The Design and Fabrication of an ARPA Network Interface

by

Richard Henry Gumpertz

Submitted in Partial Fulfillment

of the Requirements for the

Degree of Bachelor of Science

at the

Massachusetts Institute of Technology

July 1973

Signature of the Author: _____

Department of Electrical Engineering

Certified by _____

Thesis Supervisor

Accepted by _____

Chairman, Department Committee on Theses

The Design and Fabrication of an ARPA Network Interface

by

Richard Henry Gumpertz

Submitted to the Department of Electrical Engineering at the Massachusetts Institute of Technology on July 4, 1973, in partial fulfillment of the requirements for the degree of Bachelor of Science.

Abstract

This report describes the design and fabrication of a hardware interface between the ARPA Network and the Multics computer system at M.I.T. It provides a communication path from the Honeywell 6180 in one building to the ARPA Network IMP in another building about 2000 feet away and thence to computers at many different universities and other organizations currently involved in computer research.

The new interface has been accepted for production use and is expected to carry about 10% of the communications traffic of Multics. This load is expected to increase with time, as will the load due to use of other facilities offered through the network.

Thesis Supervisor: Jerome H. Saltzer

Associate Professor of Electrical Engineering

Acknowledgements

I wish to thank my advisor, Professor Jerome Saltzer, for his advice and guidance. I am especially grateful for the time he spent reviewing this report while in the midst of preparing to give a series of lectures abroad. I am also indebted to the members of the Network Group at Project MAC, Michael Padlipsky, Douglas Wells, Kenneth Pogran, and Rajendra Kanodia, who have counselled me during the stages of design and implementation. Many of the staff of the M.I.T. Artificial Intelligence Laboratory, especially Pitts Jarvis, Tom Knight, and Richard Greenblatt, helped me learn to use their PDP-10. John Williams helped substantially by preparing cables and switch panels, checking logic diagrams and wiring, and offering advice. My thanks also go to all the people at Honeywell Information Systems Inc. who helped me along the way, in particular Robert Scriver, who, in addition to giving plentiful advice, found the board mentioned in section 4.2.

Finally, I wish to thank Linda Johnson, who persevered.

This research was supported in part by the Advanced Research Projects Agency of the Department of Defense under ARPA Order No. 2095, monitored by ONR Contract No. N00014-70-A-0362-0006.

Table of Contents

1. Introduction

One of the more exciting developments over the past few years for computer researchers has been a project known as the "ARPA Network" [Rob 1]. Sponsored by the Advanced Research Projects Agency (ARPA), it consists of a series of Interface Message Processor (IMP) mini-computers interconnected by high speed leased telephone lines in a "store and forward" message switching communications network. Each of the IMP's acts as a communications processor, accepting, transmitting and delivering messages to "host" computers (which are generally medium to large scale machines) or special terminal controllers (which can be used remotely to "log in" to one of the hosts) [Hea 1]. Included among the sites for IMP's are many universities and other organizations currently doing computer research in the United States. Sites in Great Britain and Norway are expected to soon join the network. The network, currently providing the ability to "log into" other systems, to transfer files between systems, etc., has greatly increased the interaction (some of it cooperative!) between the various research groups. Further facilities currently under development promise to provide more extensive sharing of resources, thereby allowing one to use the most effective hardware for a given problem.

One of the larger computing systems connected to this network is the Multics system at M.I.T. [MAC 1]. Currently implemented on a Honeywell (formerly General Electric) model 645, this system is being switched over to the Honeywell 6180. Since

the original interface was designed to work with a "local host interface" protocol [BBN 1], which can be used for distances up to 50 feet from the IMP, the current interface will become unusable with the 6180 which is about 2000 feet away. Although this problem could probably have been solved by modifying the old interface (known as the GE-645 to IMP Special Interface or GIMPSPIF [Bhu 1]), the design and construction of a new interface offered a chance to solve several other problems at the same time.

This decision resulted in the design and construction of an Asynchronous Bit Serial Interface (ABSI) which not only meets the essential requirement of working with a long cable but also meets several other important criteria which are explained below.

This interface has been accepted by the Project MAC group responsible for Multics network development for production use as the connection of Multics to the ARPA Network. In addition, the interface is being considered for inclusion in several other Honeywell installations, both of Multics and of other 6000 series systems.

The development of the ABSI involved five principal tasks:

(1) Functional specification
(2) Logic Design
(3) Construction
(4) Debugging
(5) Documentation

In the sections which follow I describe the methods used to accomplish each of these tasks. My emphasis will be not so much on the actual device built but rather on the techniques and

principles used; these are the more interesting insights found during the work. Details on the operation of the logic may be found in the appendices to this report.

2. Functional Specification

Probably the most significant factor in the design of the ABSI was the our experience over the past few years with both the original interface hardware and the Network communications software. This experience brought about many insights/hindsights not apparent when the GIMPSPIF was constructed.

This knowledge was combined with the requirements given in two interface specifications, one for each "end" of the ABSI. The primary reference document for the IMP end is [BBN 1]. Several different possibilities were considered for the Multics end of the connection. From these, the Common Peripheral Channel on the Input/Output Multiplexor (IOM) was chosen. (This is the same type of interface used by the GIMPSPIF.) The specification for this interface is [HIS 1].

The principal improvements of the ABSI over the GIMPSPIF (which are expanded upon below) may be summarized as:

(1) Capable of using a "distant host interface" protocol

(2) Full duplex communication

(3) Implementation using standard 6000 series hardware components

(4) Operational similarity to other Honeywell peripheral devices

It has retained the following features from the GIMPSPIF:

(1) Use of the Common Peripheral Interface

(2) Functional simplicity

2.1 Simplicity

A further consideration in the functional specification of the ABSI was simplicity. It is generally a reasonable assumption that simplicity of design leads to reliability, understandability, and a cleaner implementation. It is also felt by many that hardware should be as flexible as possible, leaving to software any functions which it can reasonably handle. Once a special feature is put into hardware, it becomes difficult to change. If at some later time a slightly different action is wanted, the cost to change the feature is much greater than if it were implemented in software. These two principles led us to make the program-controlled interface to the ABSI quite simple. Almost all "bells and whistles" which could possibly be implemented in the software were eliminated from the hardware design, thereby not only reducing the number of potential "bugs" in the hardware but also allowing future changes in device control strategy to be made without modification of the ABSI.

2.2 The Common Peripheral Interface

The GIMPSPIF utilized a Common Peripheral Interface (CPI) protocol to Multics. With its 6000 series, Honeywell introduced a new interface called the Peripheral Subsystem Interface (PSI). They also introduced a special adapter which effectively looks like a memory port -- the Direct Interface Adapter (DIA). These

were considered for use with the ABSI but were rejected.

The Peripheral Subsystem Interface is really designed to communicate with a Micro-Programmed Controller (MPC). The IOM, therefore, assumes quite a bit of intelligence in the machine to which it is talking. The extra complexity in the ABSI which would have been required was considered unjustified. A new device, the Unit Record Controller (URC), is a special MPC designed to run peripherals such as line printers, card readers, etc. Because of its newness, however, the interface between it and the peripheral seemed to be a moving target. Furthermore, M.I.T. is not currently using it for any other peripheral.

The Direct Interface Adapter has a similar requirement of an intelligent peripheral. It is generally used for connection to communications processors which access the Multics memory directly.

The Common Peripheral Interface, on the other hand, specifies that the IOM will do most of the common operations such as address computation. All that the device must handle is the command sequence, the decoding of commands, and the actual data transfers (one character at a time).

2.3 Distant Interface

The first of the four improvements, use of the "distant host interface", was absolutely essential. Without it the 6180 would be unable to communicate with the network because of its physical separation from the IMP. (Actually, there were two alternatives.

A protocol known as a "very distant host interface" was seriously considered but was rejected due to lack of experience with it at any other site and its inherent complexity. A new IMP was also considered, but the lead time for installing it was far too great. Due to an increased need for network ports, a new IMP will be obtained, but not until well after the ABSI is installed.) Since the local interface could easily be implemented using the same circuitry as the distant, except for the drivers, both options were made available. The IMP cable is attached to one of two connection points depending on which interface is desired. No modification of the ABSI is necessary since it automatically determines which cable is connected and enables the appropriate set of receivers.

2.4 Full Duplex

The GIMPSPIF was implemented as a half duplex device because this required only one IOM channel rather than two, as are required for full duplex. (1) At the time of implementation, half duplex interfaces were supported by Bolt Beranek and Newman, the contractor responsible for the IMP and its software. The additional expense for a second common peripheral channel could not be justified, because the single channel could provide more

---

(1) The term "full duplex" is used to describe communication paths which can be used in two directions simultaneously. The term "half duplex" also indicates bidirectional communication, but in this case the path can be used in only one direction at a time.

than adequate bandwidth. Due to various deadlock conditions encountered, Bolt Beranek and Newman has since decided to disallow half duplex interfaces. They state:

> "Those few hosts which originally implemented half duplex interfaces have had inordinate difficulties of various kinds. [Therefore] the special interface must be operated in a full duplex manner." (2)

2.5 Standard Components

The GIMPSPIF ran into the problem that often happens when responsibility for equipment maintenance has been split among different parties -- the "blame the other guy" syndrome. Whenever hardware problems were encountered in the communications path between the Network and the Multics software, there was often much debate as to whether the problem was in the GIMPSPIF (maintained by Project MAC) or the Common Peripheral Channel (maintained by Honeywell). Although the close relationship between the two groups made such difficulties not insurmountable, it was felt that it would be preferable to turn complete maintenance responsibility over to a single organization.

The most desirable method for transferring such responsibility would have been to have the new IMP-Multics interface designed, built, supplied, and maintained by Honeywell. Inquiries regarding this possibility led to estimates of at least one year lead time and, possibly, high development costs. Since this seemed unacceptable, Project MAC decided to build the

---

(2) [BBN 1], Page 4-1

device. Nevertheless, in an attempt to facilitate maintenance by Honeywell field engineering staff, we tried to follow their design style. To further this goal, Honeywell agreed to supply parts and technical advice in return for the right to use and market the design of the interface. The design engineers and field engineers proved to be extremely helpful.

2.6 Operational Similarity

In addition to making the ABSI similar to Honeywell hardware for purposes of maintenance, it was desirable to make the programming characteristics as similar as possible to those of other Honeywell peripheral devices. The command and status codes used by the GIMPSPIF did not follow the format prescribed in the Honeywell standard for the Common Peripheral Interface [HIS 1]. As a result, the software had to handle complaints generated by the standard hardware because of this mismatch. To avoid such problems, the ABSI meets all of the appropriate specifications provided by Honeywell. Several functions omitted in the earlier interface such as parity error detection and the ability for either the channel (IOM) or the peripheral (ABSI) to initiate the termination of a data transfer were included. A programmer (3) or field engineer familiar with using any other peripheral should have no trouble understanding the IMP interface. Similarly, any

---

(3) On Multics this would normally include only system programmers, due to the general device independence offered by the operating system.

general I/O programs which handle a multiplicity of devices should be able to interpret status from the ABSI at least to the level known as "major status". (The rest of the status, "substatus", is specifically defined to be device dependent.)

3. Logic Design

Once it was known how the new interface should perform, the actual logic design was done. Although the circuitry resembles a typical teletype controller upon brief inspection, its totally asynchronous nature forced much greater consideration of potential races and other timing problems. The ABSI must, in effect, communicate with four different devices (two separate IOM channels and the two sides of the IMP) and always behave in an expected manner. These problems were solved by using conservative logic design and much hand simulation of possible input sequences from both sides of the interface.

The circuitry for the ABSI was purposely designed to not depend on using the logic gates at their maximum speed. This was possible because the two interfaces (IOM and IMP) are such that timings are on the order of about a microsecond while the gates used have propagation delays on the order of nanoseconds. It is believed that either a substantial increase or decrease in the speed of any or all of the gates would not adversely affect the operation of the ABSI. Delay lines and delay gates are used where specific timing characteristics are required.

4. Construction

Once the design of the ABSI was complete, it was possible to go on to the stages of construction, debugging, and use. It is interesting to note that more than nine tenths of the time spent on such a project is spent on such tasks as preparing drawings, obtaining parts, and fabricating the device. Any tools which can be used to expedite these jobs can significantly reduce the total time between conception and completion of implementation.

4.1 Automated Design Aids

Through cooperation between the M.I.T. and Stanford University Artificial Intelligence Laboratories (using the ARPA Network!), a powerful set of programs has been put together for hardware design. Although they do not directly simplify logic design, they greatly aid in the "clerical" work which can almost totally inundate a hardware designer. This consists of logic diagram production, generation of connection lists, checking of gate loading, checking for missing inputs or outputs, optimization of wire routing, preparation of cross-reference lists, and, finally, generation of wire lists for actual wiring of the board.

The "drawing" program allows one to edit logic diagrams in a highly interactive manner using a keyboard and a retreshed display screen. Facilities are provided for defining gates and for adding deleting, and moving both gates and wires. To ease repetitive operations, macro and "set" manipulating capabilities

are present. Text may also be put in the drawings, either as comments or as labels on signals. When complete, the drawings may be plotted using a CalComp plotter.

Once a set of drawings has been completed, one may automatically produce a signal cross-reference list directly from the drawings. At the same time, all gates are checked for errors such as overloading, paralleled outputs, and so on. In addition to its use as direct documentation, this list may be used as input to the routing program which optimizes wire runs and prepares directions for the person who will do the wiring. Although facilities are also available for preparing paper tapes for automatic wire-wrap machines, these were not used.

These programs did require a large investment of time to learn how to use fully. In addition, several modifications to the programs were necessitated by the special requirements of Honeywell logic and board layout. Although this is a one-time investment, there are currently no plans for any similar hardware construction, so this time could not be amortized over several different projects. It should be pointed out, however, that documentation is currently far beyond that which would exist if automatic aids had not been applied. The drawings are both up-to-date and neat, a situation which is rarely true for prototype hardware (except, possibly, when draftsmen are availiable). The cross-references will be brought up to date simply by re-executing the programs which produced them. I feel that the extra time invested in the use of automatic design aids

has more than paid for itself in this one project alone. A test message generator used to check out the read channel, for instance, required only two hours for production of the drawings, cross-reference lists, and wire lists. In my opinion, these programs are at least as significant a step forward for hardware development as higher-level languages were for software development.

4.2 Physical Characteristics

The actual construction was done on a single board with room for 200 small and medium scale integrated circuits in 14-pin dual in-line packages (DIP's). The circuitry is mostly Honeywell 500 series logic, similar to the Sylvania SUHL II line. Interconnection is through wires hand wrapped on the reverse side of the board. This board, dug out of a locker at Honeywell in Phoenix, was much more suitable for my purposes than the boards currently being used for production work, as it included plug-in sockets for the DIP's, rather than soldered connections. Although less reliable on a long-term basis, this setup is far preferable for prototype production, which must be done by hand. It is expected that any further ABSI's will be constructed using the more permanent arrangement.

A few out of stock parts, which were not available from local commercial suppliers either, were created using discrete components mounted on a 14-pin DIP pad. (This was possible because they all happened to be resistor/capacitor chips.) They

may be replaced, without any wiring changes, as soon as normal parts become available.

5. Debugging

Careful planning of the implementation schedule minimized the time between construction and successful operation of the ABSI.

In particular, the project was built and tested in an incremental manner. The steps were:

(1) Construct and test (off-line) the write side of the interface in the "local interface" mode.

(2) Connect the write side to the IMP and attempt to send messages from the 645 to the teletype attached to the IMP.

(3) After incorporating any changes found while debugging the write side, construct and test the read side.

(4) Test the new Multics software using the "local interface".

(5) Switch to the "distant interface" for tests from the IOM maintenance panel and then from the Multics software.

(6) Move to the 6180 for testing over the long cable.

Since much of the logic for the write side of the ABSI is very similar to that of the read side (in particular those parts which accept commands from the IOM and return status to it), it was obvious that one of them should be built before the other. This would allow any changes found in the first side to be incorporated into the second. Since the only connections between the two sides are for initialization, host up/down, and IMP up/down signals, this incremental manner of construction was fairly easy.

5.1 Methods of Testing

Since there were no spare ports on the M.I.T. IMP, any testing done with the IMP required disconnecting the current Multics-IMP interface, and with it the "service" system. Since there are currently many users at a great distance from M.I.T. who use Multics solely through the network, such test sessions were minimized so as to infringe on them as little as necessary. Instead, two other testing methods were developed.

The first, usable even when only the write channel existed, was simulation of the IMP. A circuit which simply accepts any message sent to it was built. A facility for generating messages, although a bit more complex, was also easily implemented. Note that both of these circuits were built on the same board as the rest of the ABSI so that they may be retained for use in maintenance, in addition to their original debugging purposes.

The protocol between the ABSI and the IMP is such that it is possible to connect the IMP end of the write side to that of the read side (i.e. "back-to-back"). This allows messages to be sent and read back for verification. Although used only briefly for the original debugging, this method of testing will probably be employed by "test and diagnostic" programs used by the field engineers.

Like the IMP port, the central processors and memory are also a valuable resource, and so are usually run on the "service" machine. The IOM is not normally used. (There are two I/O

controllers available but one is sufficient for the load on the "service" system.) Thus any testing which can be done using only the IOM is preferable because it is using hardware that is often idle. Due to the presence of a maintenance panel specifically designed for use in checking out hardware, such testing was not only easy but also often preferable to using software.

Initial testing was done on the 645 for several reasons. The most important was its physical proximity to the IMP. This not only allowed the simpler "local host interface" to be used, but also allowed one to conveniently walk back and forth between the IOM, the ABSI, and the IMP. In addition, the 645 seemed to be more easily available than the 6180. I had the 645 IOM virtually to myself; the 6180 was undergoing testing and checkout and so was required by several different people trying to debug software, hardware, or both. Finally, the cable betwen the two buildings was not ready for use in time for the testing of the ABSI.

5.2 The Write Side

The logic of the write channel is inherently simpler than that of the read channel for several reasons. The most obvious is that all data transfers, including command sequences, are in the same direction: IOM to ABSI. Thus only one data butter is required to hold either command codes or message data characters. Furthermore, simulation of the IMP on the write side can be done with a single inverter since one can use an oscilliscope to

observe the data being sent to the pseudo-IMP. The read side, on the other hand, must have some sort of simple pattern generator built into its IMP simulator. Otherwise one cannot spot data dependent hardware bugs such as errors in shifting.

The write side is also simpler in that it always terminates normal dat transfers when told to do so by the IOM. The read channel must terminate when told to do so either by the IOM or the IMP. If told to terminate by the IMP, there is a high probability that it will have to pad the data to form a full character, thus adding additional complexity. For these reasons, the write channel was chosen as the first side to implement.

Initial checkout produced a few problems with the command interpretation logic. One nasty problem to track down was a timing error when a WRITE command had been issued. Once found, however, this, along with several more trivial bugs, was fixed without too much difficulty.

Once the write side was able to execute all of its commands and data transfer seemed to be working (as verified on an oscilloscope), the interface was connected to the IMP. The IMP simulator used to debug the write side was connected to the IMP to simulate the read side of the ABSI. In this way the messages generated by the IMP for the host were read and discarded. (Had this not been done the IMP would have considered the host as "down" and would have discarded messages sent to it by the host.) Using the IOM maintenance panel, test messages were sent addressed to the teletype attached to the IMP, with a bit set to

specify that they were to be printed in octal. When this had been successfully done, we connected the IOM to a memory box and a 645 central processor. Using a special program developed for testing channels, (4) we sent alphanumeric messages to the IMP's teletype. Satisfied that data was being transmitted correctly, we did not attempt communicating with any other hosts on the network. This would have involved protocols which seemed too complex to justify hand simulation. We were confident enough in the logic to wait for the read side to be completed for further testing.

5.3 The Read Side

After including the changes to the command sequence logic found necessary in the write side, I constructed the read side. A very simple (consisting of a single jumper!) IMP simulator for this side (which would send a message of indeterminate length consisting of all zeros) was also constructed. The command sequence logic, due to its prior debugging on the write side, was operational about two hours after it was first tried. The errors found were not design errors but rather one wiring error and one typographical error in the schematic drawing which caused a gate input to be connected to the wrong signal. The logic responsible for reading messages from the IMP seemed to work on its first

---

(4) TSTCHN, which runs under BOS, a simple operating system used for debugging the hardcore (e.g. virtual memory management) portions of Multics itself.

trial using the simple message generator. At this point a new IMP simulator was designed and constructed which would generate more interesting test messages. Based on a modulo 40 counter, it alternates between two different messages which were designed to check the obvious potential weak points of the ABSI, including such things as padding messages to character boundaries, single bit messages, various data patterns, proper positioning of bits within the message (i.e. no extra or missing shifts), etc. In addition the messages were designed so that they could be quickly verified in the lights of the IOM maintenance panel. The pattern generator proved very valuable in that it showed up several bugs in the design. The first was that one bit messages were not being correctly read. Although all messages are currently at least 16 bits long, this assumption clearly should not be built into the hardware. Another bug, which was simply a design omission, involved the padding of messages. As originally implemented, the message was padded with bits equal to the last bit of the message; it should have been padded with zeros. Both of these problems were found and eliminated by the end of the day.

5.4 Final Debugging

Because of a series of several hardware failures, there was significant delay in obtaining a "development" system (5) for

---

(5) The Multics system at M.I.T. normally runs with two central processors on the "service" machine. When needed for testing hardware or debugging a software change that may crash the system (such as a change to the user scheduling algorithm), a processor,

running the ABSI with the new network software. Work on testing the distant interface was, therefore, begun ahead of schedule; no problems were encountered.

The software for controlling the new interface, written by Rajendra Kanodia, did not work at first due to a bug in the PL/I compiler. After writing around this, however, it also appears to work without problem.

As of the time of writing, the interface has not been tested on the 6180 system. This is principally because the cable between the two buildings is not yet ready. In addition, the 6180 has frequently not been available for test sessions because of high priority hardware modifications being installed. We expect to start these tests very soon. No potential problems, other than trivial problems such as determining where to install the ABSI, are foreseen.

---

some memory, and a disk channel can be reconfigured out of the system (without stopping it!) to form a separate "development" system.

6. Documentation

One of the most important considerations in the success of any project of this sort must be the documentation produced. Unless fully explained, a piece of hardware can become virtually useless. It is believed that the preliminary design documents and the block level descriptions of operation, together with the drawings, signal cross-reference lists, and the wire routing lists (attached as appendices) are quite sufficient to allow one to understand and repair the ABSI should any problems arise. The preliminary design document, which also serves as a programming reference, is attached as an appendix. The automatic programs used to produce the latter three documents help guarantee the accuracy of these documents, and so their usefulness. It is doubtful that such documents would be nearly as accurate as they are (They might not even exist!) without these tools.

7. Conclusions

This report has discussed some of the thoughts behind the design of the ABSI. The real conclusion of the project cannot be expressed on paper; it is the interface itself. The final test of its success will be its around-the-clock usage by customers throughout the country.

Bibliography

[BBN 1] ---, Specifications for the Interconnection of a Host and an IMP, Report No. 1822, April 1973, Bolt Beranek and Newman Inc., Cambridge, MA

[Bhu 1] Bhushan, Abhay, "The GE645-IMP Special Interface", M.I.T. Project MAC Computer Network Group, Memorandum No. 3, 7 August 1970

[Hea 1] Heart, F.E., Kahn, R.E., Ornstein, S.M., Crowther, W.R., and Walden, D.C., "The interface message processor for the ARPA computer network" Proceedings, Spring Joint Computer Conference, American Federation of Information Processing Societies, May 1970

[HIS 1] ---, Product Performance Specification, Common Peripheral Interface, Document 43A130524 Revision H1, Honeywell Information Systems Inc., Phoenix, AZ

[HIS 3] ---, Engineering Product Specification, Part 2, Input/Output Multiplexor (IOM) Honeywell Information Systems Inc., Phoenix, AZ

[MAC 1] ---, Multics Programmers' Manual, Revision 14, 30 April 1973, M.I.T. Project MAC, Cambridge, MA

[Rob 1] Roberts, Lawrence G. and Wessler, Barry D., "Computer network development to achieve resource sharing", Proceedings, Spring Joint Computer Conference, American Federation of Information Processing Societies, May 1970

Appendix 1

The following document, originally circulated within the Computer Systems Research Division of Project MAC, is both the design specification and programming reference for the ABSI.

PROJECT MAC

Computer Systems Research Division

February 8, 1973

Request for Comments No. 8

ASYNCHRONOUS BIT SERIAL INTERFACE -- PRELIMINARY SPECIFICATION

by Richard H. Gumpertz

DRAFT

date 2/12/73

1. General Description

For some time it has been clear that a replacement for the current Multics interface (the GIMPSPIF) to the ARPA Network IMP (Interface Message Processor) would be necessary for use with the 6180 system.

The principal reason for this was the necessity of using a "Distant Host Interface" arrangement since the IMP is to remain at 545 Technology Square whereas Multics will reside in building 39. It has also been concluded that a distant interface will be necessary even when a new IMP is installed in building 39. In addition, it was deemed necessary that the new interface run in a full duplex mode, thereby allowing simultaneous read and write operations.

I am currently designing and building such an interface to be called the Asynchronous Bit Serial Interface (ABSI). Like the GIMPSPIF, the new interface will communicate with Multics via a Common Peripheral Interface (CPI) as defined in Honeywell document number 43A130524, and to the IMP as defined in Bolt Beranek and Newman report 1822. Unlike the GIMPSPIF, two CPI type channels will be needed to allow full duplex operation. One will be used for write operations (HOST to IMP) and the other for read operations (IMP to HOST). The new interface will use the "two way handshake" procedure (see BBN-1822) for communicating with the IMP. The timing will be made slow enough to meet distant host interface requirements but it will be capable of running with either a local or distant interface. This will allow a theoretical maximum transfer rate of about 800,000 bits/sec. in each direction. The ABSI will attempt to meet as closely as possible the CPI specifications. All known exceptions are summarized in section 5 below. It is expected that the interface will operate with any Honeywell 6000 series IOM Common Peripheral Channels.

The following description assumes that the reader is familiar with BBN-1822 and the Common Peripheral Channel specifications and operation. No attempt is made to duplicate the information contained there.

---

2. Operation

The new interface will accept the following commands on the "Write Channel":

OPCODE (octal)	MEANING
00	REQUEST STATUS
11	WRITE (start output)
20	HOST UP
40	RESET STATUS
60	HOST DOWN

The "Read Channel" will accept:

OPCODE (octal)	MEANING
00	REQUEST STATUS
01	READ (start input)
40	RESET STATUS

The device code sent as part of the command sequence will be ignored although its parity will be checked.

2.1 Request Status (00 on either channel)

This command will return the same status as the previous command on the same channel. Previous COMMAND REJECT status, however, will be ignored so that if COMMAND REJECT status is received in reply to REQUEST STATUS, it indicates that there was an error in the transmission of the REQUEST STATUS command itself. Note that the status stored may indicate the substatus IMP DOWN even if the IMP has since come up. To correctly obtain the current IMP up/down status, RESET STATUS (see section 2.5) should be used. (A similar condition exists for the substatus HOST DOWN.)

2.2 Read (01 on the read channel)

This command will start reading a message from the IMP. It will terminate under any of the following conditions:

1. The LAST IMP BIT line is received along with a data bit.
2. The host up/down relay is changed to the down state by either of the following:
   a) the manual HOST DOWN pushbutton on the interface
   b) the HOST DOWN command on the write channel

3. The IMP goes down (according to its up/down relay).
4. The channel decided to terminate the command.
   Note that terminate will not actually happen until just after a complete character (6 bits) has been transmitted to the channel. Possible causes are:
   a) The storage buffer indicated in the DATA DCW has been filled and there are no additional DATA DCW's.
   b) The channel is "masked" by the software (see the IOM specifications).
   c) Some other condition occurs which causes masking, such as a second "connect" to the channel before the first list has completed.
5. A "reset" signal is received. This may be created by the manual pushbutton on the ABSI, by disconnection of either of the IOM channels from the interface, by initializing the IOM controlling the channels, or by loss of power to either of the IOM channels. Note that all transmissions are aborted immediately and no status is stored. When the "reset" level is removed, the channel will be ready to accept a new command no matter what it previous state was. The "reset" condition also forces the host up/down relay to open (i.e., HOST DOWN).

In normal operation, only cases 1 (COMPLETE MESSAGE, no errors) and 4a (INCOMPLETE MESSAGE, no errors) should occur. If the latter occurs, the termination status will be READY with substatus INCOMPLETE MESSAGE. The message may be reconstructed by appending the data read by the next READ command to the end of the current data. Unlike the GIMPSPIF, the READY FOR NEXT IMP BIT line is not raised until a read command is issued. Thus there is no way to determine that the IMP has a message waiting until a READ command is issued. It is expected that the software will normally keep a READ command pending at all times that the system is in operation. The expected method for aborting such a read command is to issue a HOST DOWN command on the write channel. If this fails, the HOST DOWN pushbutton on the ABSI should be used. If this also fails, the RESET pushbutton on the ABSI should be tried.

2.3 Write (11 on the write channel)

This command starts writing a message to the IMP. Normally termination will be done only on end of message condition in the buffer. Note, however, that any of the conditions listed in section 2.2 above for aborting the READ command, also apply (except obviously 1 and 2b). The LAST HOST BIT level will

be sent with the last bit transmitted only if termination is caused by the channel rather than the ABSI. Also note that termination under conditions 4b and 4c (the channel becoming masked) will take place only after the next character (6 bits) has been sent to the IMP.

2.4 Host Up (20 on the write channel)

This command causes the HOST MASTER READY line to be connected to the HOST READY TEST line by closing a relay. The software must make sure that no READ or WRITE command is issued to either channel until the relay has had a chance to settle. The delay involved is not here specified, because relay specifications may vary. It is believed, however, that the programmer may assume 100 msec. to be sufficient. This command may be simulated at any time by the manual HOST UP pushbutton.

2.5 Reset Status (40 on either channel)

This command is identical to REQUEST STATUS, except that any resettable status condition from the previous command will be reset. Note that a RESET STATUS command is effectively executed before any command (other than REQUEST STATUS). Resettable status is defined as:

1. any DATA ALERT condition
2. INCOMPLETE MESSAGE substatus
3. IMP DOWN substatus if the IMP is no longer down
4. HOST DOWN substatus if the system is no longer down

2.6 Host Down (60 on the write channel)

This command will open the relay closed by the HOST UP command. If a READ operation is taking place on the read channel, it will be aborted when that channel sees the relay open. It is possible the HOST DOWN command will not take effect if followed too closely by a HOST UP command (i.e., the relay doesn't get a chance to open). Whenever the relay is open (or opening), READ and WRITE commands will be rejected with the status COMMAND REJECT and the substatus HOST DOWN. The HOST DOWN command may be simulated at any time by the manual HOST DOWN pushbutton.

3. Special Interrupts

If the host up/down relay is in the up condition when the IMP comes up (closes its IMP READY relay) then a "special interrupt" will be sent on the READ channel.

4. Status

The following status codes have been defined for the ABSI:

Major Status	Substatus	Meaning
x000		Ready
	xxlxxx	IMP Down
	xlxxxx	HOST Down
	lxxxxx	Incomplete Message
x011		Data Alert
	xxxlxx	Parity Error
	xxlxxx	IMP Down
	xlxxxx	HOST Down
	lxxxxx	Incomplete Message
x101		Command Reject
	xxxxxl	Invalid Operation Code
	xxxlxx	Parity Error in Command Sequence
	xxlxxx	IMP Down
	xlxxxx	HOST Down
	lxxxxx	Incomplete Message
lxxx		Read/Write Command in Progress

Any substatus condition may occur simultaneously with any of the other substatus conditions under the same major status. If this occurs, the appropriate substatus codes will be merged to form the substatus returned.

4.1 Ready Status

The READY status indicates that the channel is ready to accept a command and that no (unreset) errors occurred in the previous command to the channel.

4.1.1 IMP Down Substatus (READY)

The IMP DOWN substatus indicates that the IMP READY relay is open (i.e., the IMP READY TEST line is not connected to the IMP MASTER READY line.) Note, however, that once this bit comes on (i.e., the IMP goes down) it stays on until it is reset by a command other than REQUEST STATUS (e.g., RESET STATUS). Note that this relay, like the host up/down relay, is subject to bounce and so it would be a good idea to wait for the relay to settle before rechecking the status. One could also wait for the special interrupt which says the IMP is up (see section 3 above), but this may be unreliable if one does not carefully consider race conditions. A "timeout" on waiting for the interrupt would therefore be suggested.

4.1.2 Host Down Substatus (READY)

The HOST DOWN substatus indicates in a similar manner that host up/down relay has been, is, or is becoming open.

4.1.3 Incomplete Message Substatus (READY)

The INCOMPLETE MESSAGE substatus may occur only on the READ channel. It indicates that termination occurred before the LAST IMP BIT level was received. Since no bits are lost when this occurs, the entire message may be reconstructed by appending the data read by the next READ command to that read by the most recent READ command.

4.2 Data Alert Status

This status is stored if an error has occurred during the transfer of data in the appropriate direction. Any error will cause the channel to abort the current operation immediately.

4.2.1 Parity Error Substatus (DATA ALERT)

Note that the READ channel will indicate a parity error in the channel status rather than the normal status field since this error is detected by the channel rather than the ABSI. Therefore PARITY ERROR substatus can occur only on the WRITE channel. It indicates that a character has been received from the IOM channel with bad parity. The WRITE command is aborted and the LAST HOST BIT level is not sent to the IMP.

The software may thus indicate to the IMP that this message should be ignored by flashing the HOST UP line (waiting appropriate times for relay closure/opening) and then exchanging a reset sequence with the IMP. (Note that this error recovery procedure will also abort any pending read operations when the HOST DOWN command is executed.)

A much less drastic method for causing the IMP to discard the message in error would be to immediately WRITE a second message whose length is greater than the maximum message length (256 words, for instance, is greater than the current maximum length of 8096 bits). Since the previous WRITE did not have a LAST HOST BIT level sent, this second message will be appended to the first, forcing it to be discarded by the IMP. The original message may then be rewritten as if nothing had happened (assuming no further PARITY ERROR's occur).

It should be noted that the PARITY ERROR substatus normally will happen only if a logic gate fails and so it may actually be reasonable to consider such an error fatal.

4.2.2 IMP Down Substatus (DATA ALERT)

IMP DOWN substatus indicates that the IMP went down at some time during the Read/Write operation. Data may have been lost.

4.2.3 Host Down Substatus (DATA ALERT)

HOST DOWN substatus indicates that the host up/down relay become open (e.g., the HOST DOWN pushbutton was pushed) during the Read/Write operation. Data may have been lost.

4.2.4 Incomplete Message Substatus (DATA ALERT)

INCOMPLETE MESSAGE substatus is solely an indication of whether the LAST IMP BIT level was received before the operation terminated. (It occurs on the READ channel only). This information is actually of little use since data may have been lost due to the error which caused the DATA ALERT status. INCOMPLETE MESSAGE substatus will always occur in conjunction with either HOST DOWN or IMP DOWN when the major status is DATA ALERT.

4.3 Command Reject Status

COMMAND REJECT status indicates that the command could not be accepted for some reason.

4.3.1 Invalid Operation Code Substatus (COMMAND REJECT)

INVALID OPERATION CODE substatus indicates that the command code received by the ABSI was not one of the legal commands for that channel.

4.3.2 Parity Error in Command Sequence Substatus (COMMAND REJECT)

PARITY ERROR IN COMMAND SEQUENCE indicates that either the device code (which is otherwise ignored) or the command code received by the ABSI had bad parity. The command is not executed even if the code is legal.

4.3.3 IMP Down Substatus (COMMAND REJECT)

For all operations except READ and WRITE, this substatus can occur only in conjunction with one of the above two command transmission errors. In this case this status is described in section 4.1.1 above.

For READ and WRITE operations, this substatus indicates that the command could not be initiated because the IMP READY relay is open. Due to timing considerations in the ABSI, the fact that this status is returned does not indicate that the IMP is still down. Therefore, one should do a RESET STATUS command to verify that the IMP is really down. If the RESET STATUS command indicates that it is up, the READ or WRITE command should be reattempted.

4.3.4 Host Down Substatus (COMMAND REJECT)

This substatus is analogous to IMP DOWN substatus in section 4.3.3 above.

4.3.5 Incomplete Message (COMMAND REJECT)

This substatus can only occur in conjunction with one of the two command transmission errors in section 4.3.1 and 4.3.2 above. It is otherwise analogous to the substatus of the same name, described in section 4.2.4 above (and is likewise nearly useless). Note that in this case it refers to the previous READ/WRITE operation, not the rejected one.

4.4 Read/Write Command in Progress

This status indicates that a READ/WRITE command was properly received. It may be returned together with the termination status after a READ/WRITE command has completed (either normally or abnormally). This status is often called CHANNEL/PERIPHERAL SUBSYSTEM BUSY.

5. Exceptions and Assumptions

This section describes all the known assumptions concerning the interfaces with the CPI type channels and with the IMP. Also noted are any exceptions to the specifications.

5.1 Restriction on EDT Transmission on the Write Channel

The WRITE channel must know when it is writing the last bit to the IMP so that it can assert the LAST HOST BIT level. In order to avoid extra buffering and a large increase in complexity, it is necessary that the EDT strobe arrive no more than 4 μsec after the WRITE CLOCK TO PERIPHERAL strobe. This is in spite of the section on Write Command Termination (section 4.3.3, sheet 32) of 43A130524. Honeywell document 43A177715 (the IOM specifications) section A2.3.4.3 states, however, that the IOM CPC sends the EDT .5μsec after the WRITE CLOCK TO PERIPHERAL and so there should be no problem. Since the ABSI is intended for use only with the 6000 IOM, no problems are foreseen.

5.2 Restriction on EDT Transmission on the Read Channel

The READ Channel must know whether the most recent character sent to the channel is to be the last character before it can read the next character from the IMP (because we have decided to eliminate read-ahead which is an unjustifiable complexity). As defined in 43A130524 the EDT strobe may arrive an arbitrary time after the READ CLOCK TO PERIPHERAL strobe. Clearly the ABSI must make some restriction on how long it will wait before reading the next character. For this reason a delay gate with delay time n is used.

The current IOM Common Peripheral Channel villates the standard in 43A130524. It sends the EDT instead of the last READ CLOCK TO PERIPHERAL, rather than after it. The ABSI will work equally well with this situation, and in fact could be simplified. The delay gate has been retained, however, for compatibility, although the capacitance has been set to give a delay time of roughly n=0. Thus, if the channel is changed later to meet the standard, the ABSI may be made compatible simply by changing the capacitance to give a reasonable waiting time.

5.3 Lack of Transfer Timing Error Detection

In spite of section 6.2 (sheet 52) of 43A130524, no provision is made for detecting TRANSFER TIMING ERRORs. This is because the asynchronous nature of the ABSI makes such detection unnecessary; its inclusion would only add nearly useless logic to the interface.

5.4 Definition of Data Transmission Time

In many places 43A130524 specifies times as "sufficient time ...to set ... receivers". It is assumed that 1μsec. is more than sufficient to cover all skew, cable length, and settling time considerations.

5.5 Minimum Buffering Requirements

Section 6.5 of 43A130524 specifies a minimum buffering capability of two characters for the WRITE channel. Only one is provided. Since the ABSI is asynchronous, no problem except a very slight speed loss should be encountered.

5.6 Maintenance Requirements

Section 6.6 of 43A130524 is not met in that neither a Running Hour Clock nor full off-line test facilities have been provided. It is expected that at least an IOM will be available for testing (via the IOM maintenance panel).

5.7 Voltage Levels for the Distant Interface

BBN-1822 specifies ± .5 volt signals for the distant interface connections; the drivers used are closer to ± .75 volt. BBN indicates that this should cause no problem (and, in fact, is probably better).

6. Physical Characteristics

The ABSI is implemented on a single 12" x 12" Honeywell MQX type circuit board. Communication with the two CPI channels is through the two free-edge connectors (one channel on each connector). Communication with the IMP is through the backpanel, using slip over connectors on the backpanel wire-wrap pins. It is required that -5v. be provided to the board through backpanel in addition to the +5v. and ground normally provided. -5v. is not normally connected to the backpanel although it is available in the power supply. A standard wire exists for making the connection. It is expected that the ABSI will work in any slot providing such power and backpanel connections. In particular, it may be mounted in either a 6000 IOM or a DataNet 355 IOM payload slot.

The logic used is Honeywell 500 series (similar to Sylvania SUHL II) with the exception of the distant interface drivers and receivers which are 400 series logic and are similar to the TI SN75108-110 group of integrated circuits.

Appendix 2

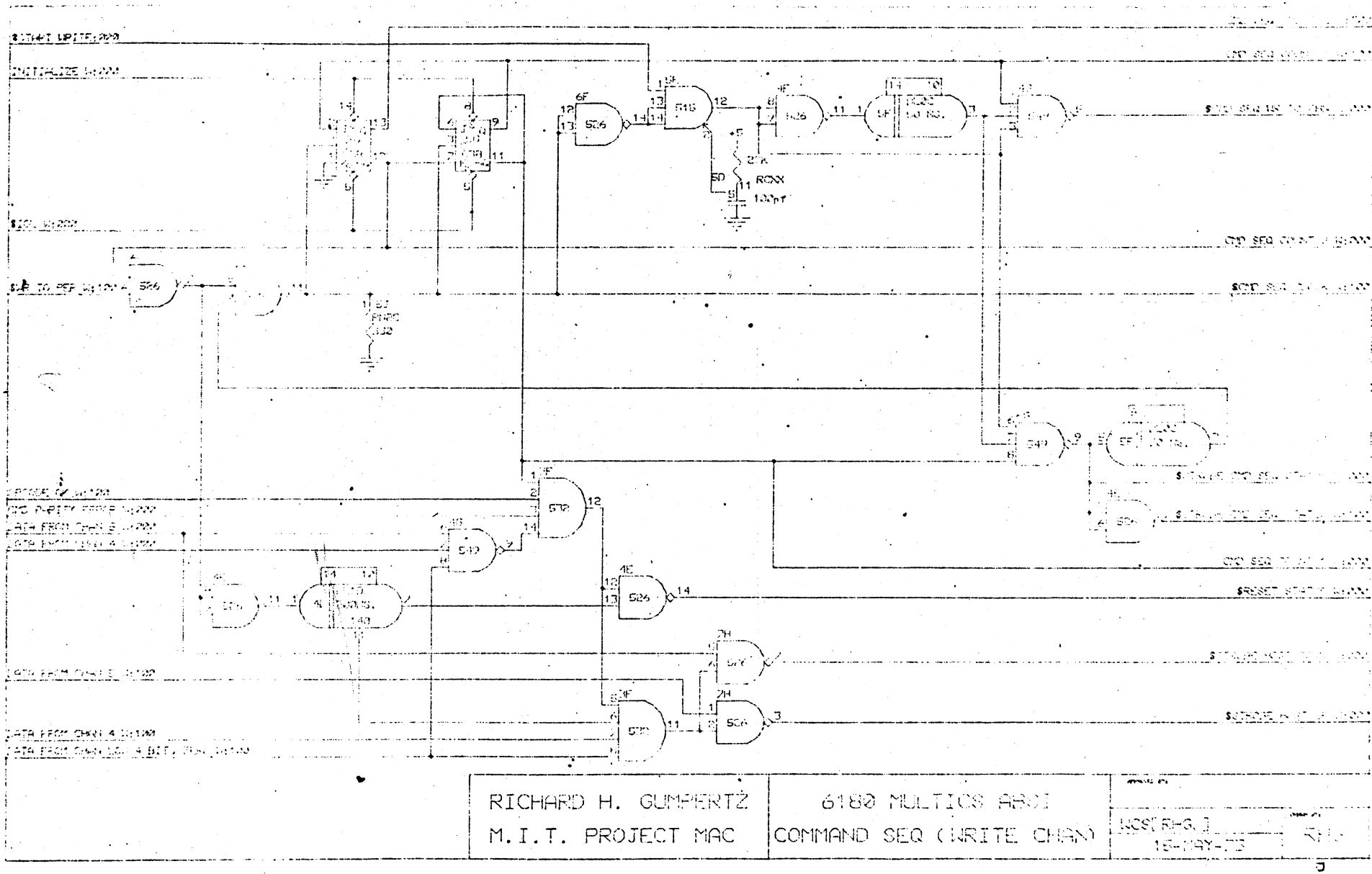
The following pages are the machine-plotted logic diagrams of the ABSI. They have been photographically reduced from their original 11" by 17" size.

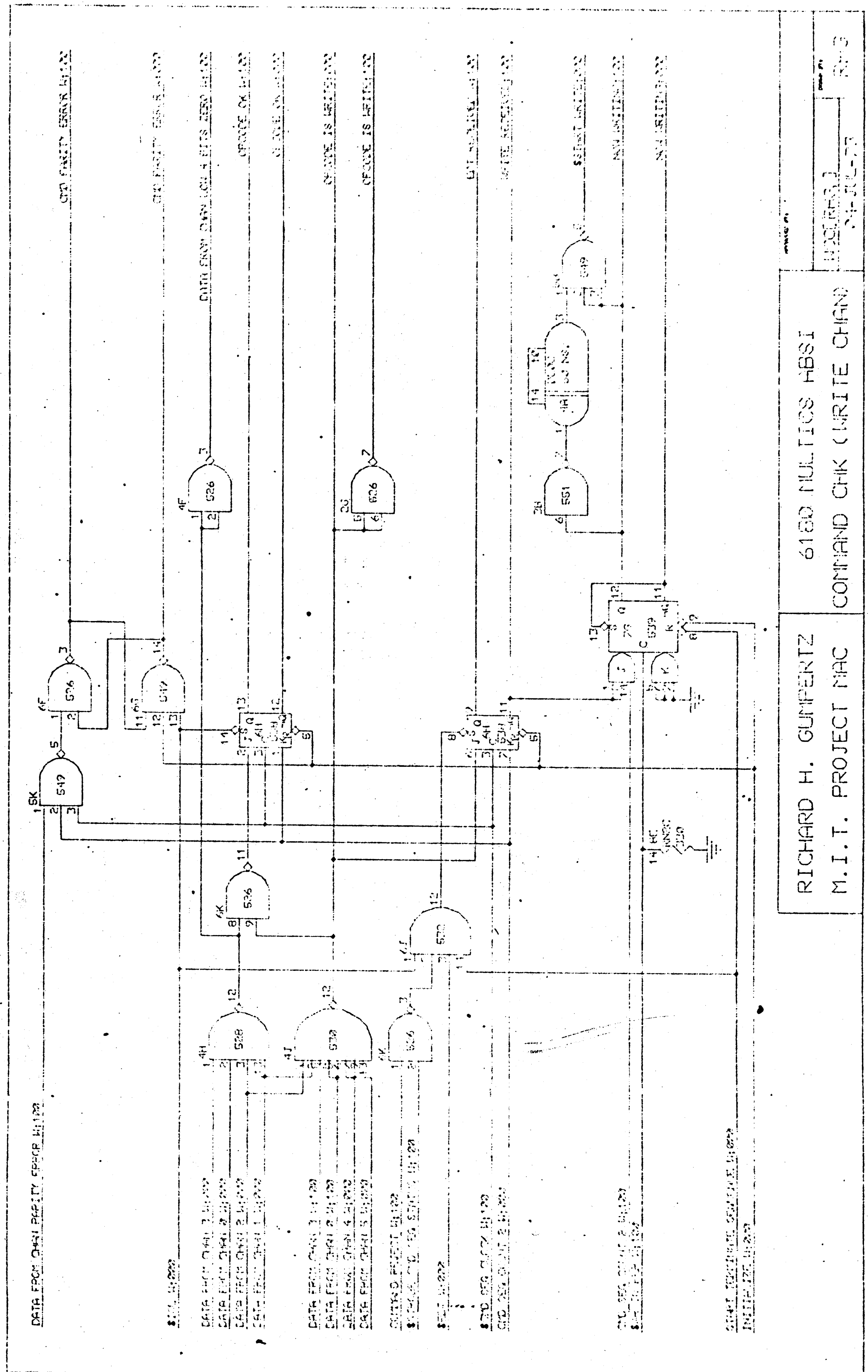
6180 Multics ABSI
(Asynchronous Bit Serial Interface)

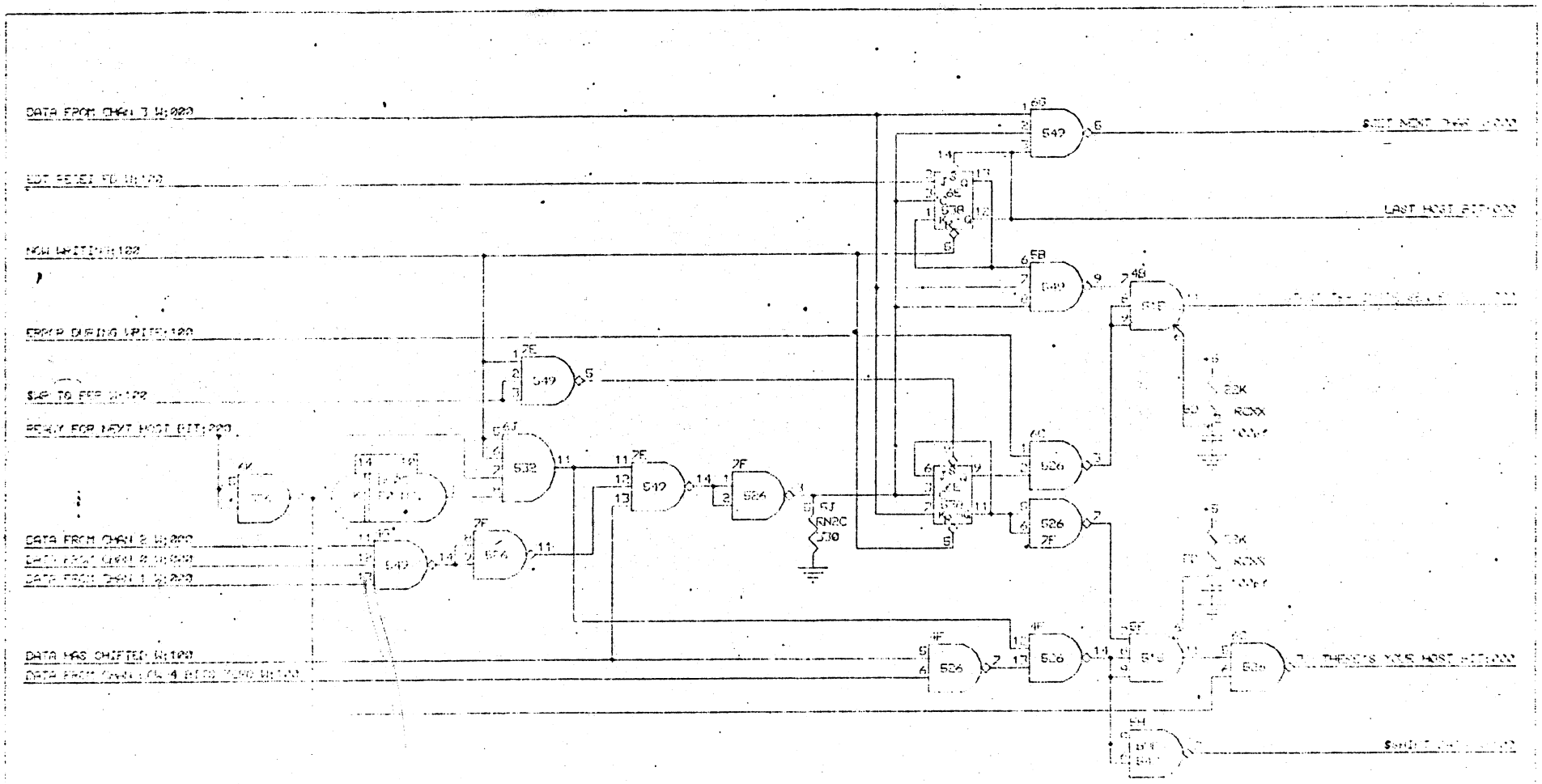
For connection to the ARPA Network

Richard H. Gumpertz
M.I.T. Project MAC

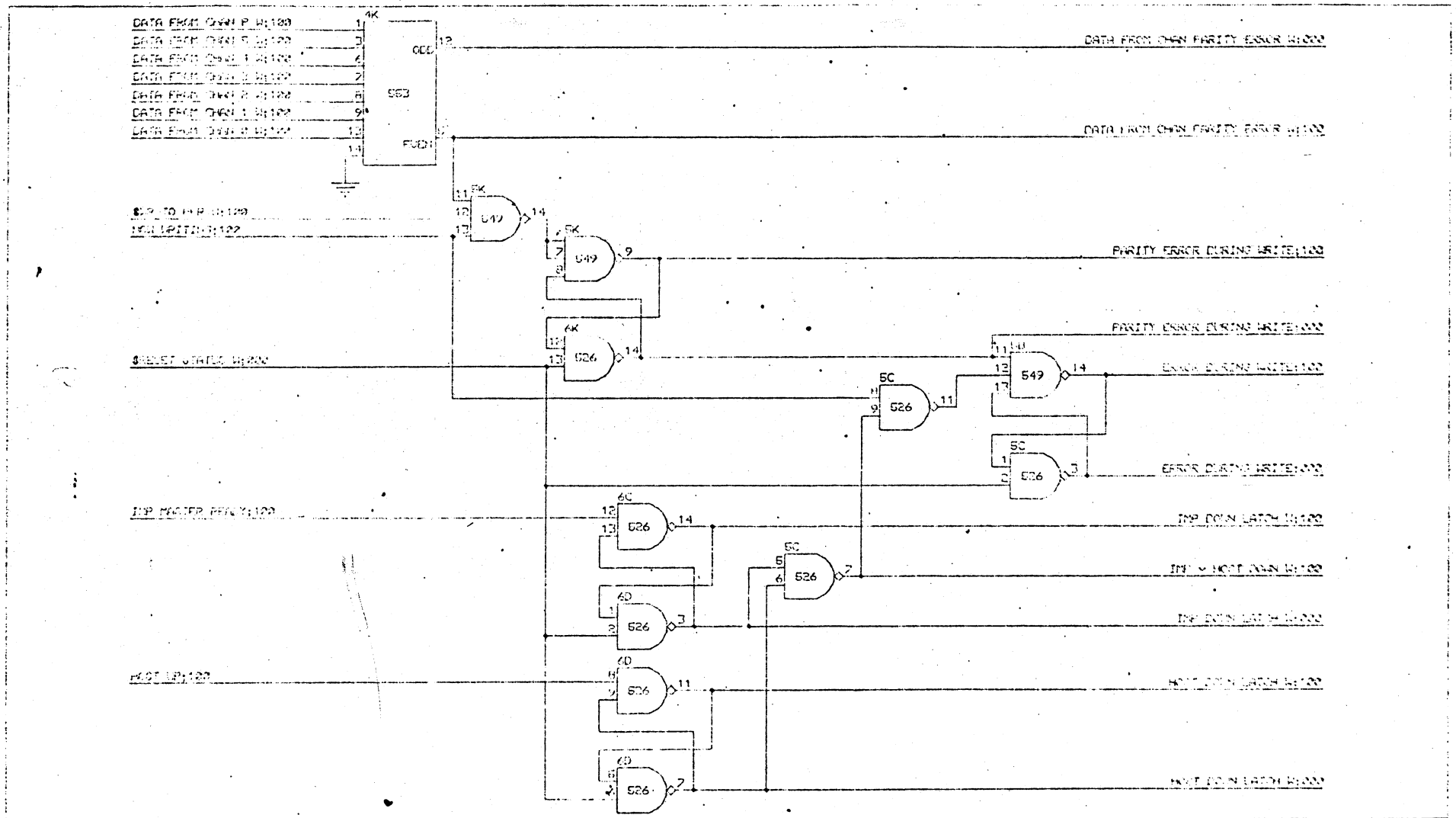
RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI TITLE SHEET	TITLE[RHG,] 11-MAY-73	RHG

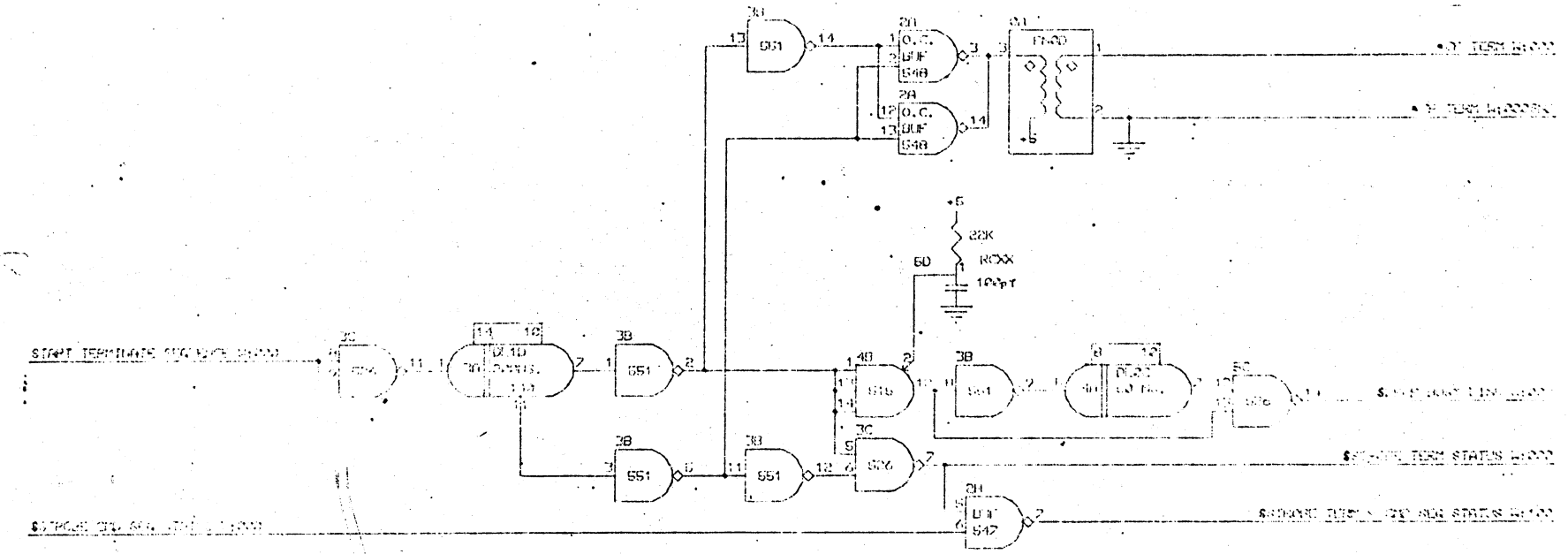

RICHARD H. GUMPERTZ
M.I.T. PROJECT MAC
6180 MULTICS ARCI
COMMAND SEQ (WRITE CHAN)


RICHARD H. GUMPERTZ
M.I.T. PROJECT MAC
6180 MULTICS HBSI
COMMAND CHK (WRITE CHAN)

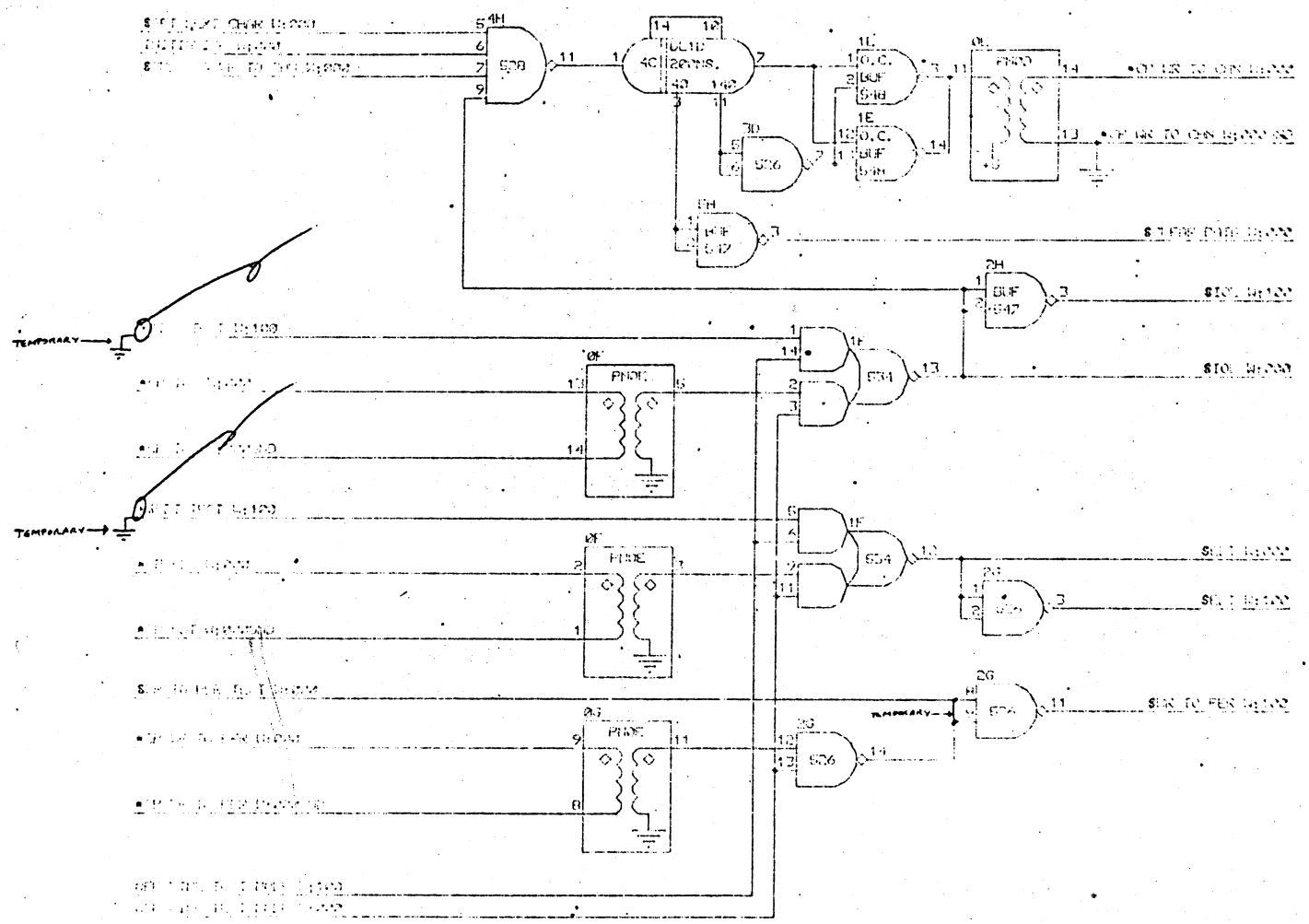


RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI IMP CONTROL (WRITE CHAN)	

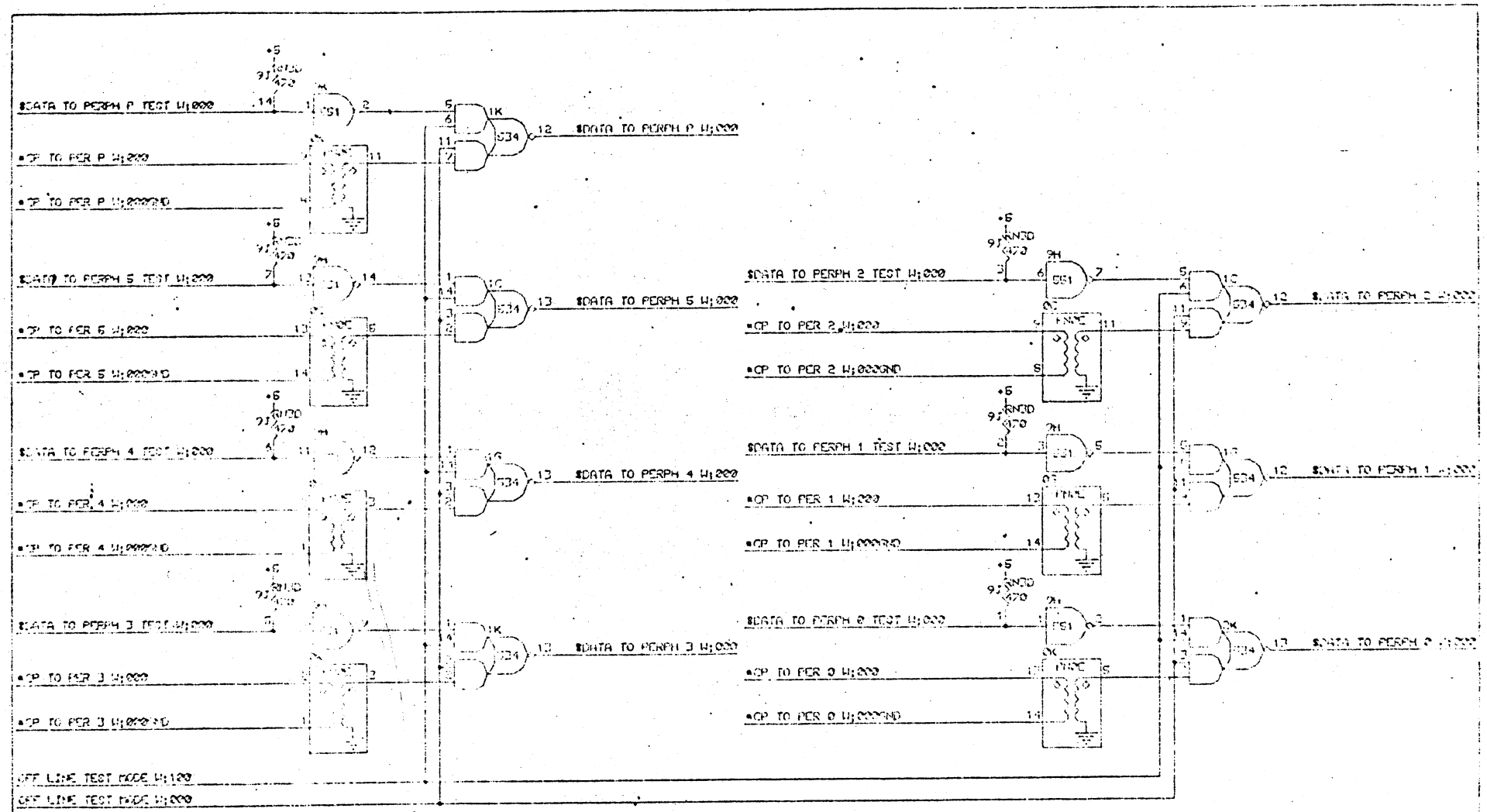

DATA FROM CHAN P H:100
DATA FROM CHAN 5 H:100
DATA FROM CHAN 4 H:100
DATA FROM CHAN 3 H:100
DATA FROM CHAN 2 H:100
DATA FROM CHAN 1 H:100
DATA FROM CHAN 0 H:100
4K
963
ODD
PUSH
DATA FROM CHAN PARITY ERROR H:000
DATA FROM CHAN PARITY ERROR H:100
$CP TO IOP H:100
IOM WRITE H:122
5K
549
PARITY ERROR DURING WRITE H:100
PARITY ERROR DURING WRITE H:000
$SELECT STATUS H:200
4K
526
5C
526
549
ERROR DURING WRITE H:100
5C
526
ERROR DURING WRITE H:000
IOP MASTER PFLG H:100
6C
526
IOP DOWN LATCH H:100
5C
526
IOP ~ HOST DOWN H:100
4D
526
IOP DOWN LATCH H:000
HOST UP H:100
4D
526
HOST DOWN LATCH H:100
4D
526
HOST DOWN LATCH H:000
RICHARD H. GUNPERTZ
M.I.T. PROJECT MAC
6180 MULTICS ABSI
ERROR CHECK (WRITE CHAN)
WECC RHS 1
26-JUN-73
RHG



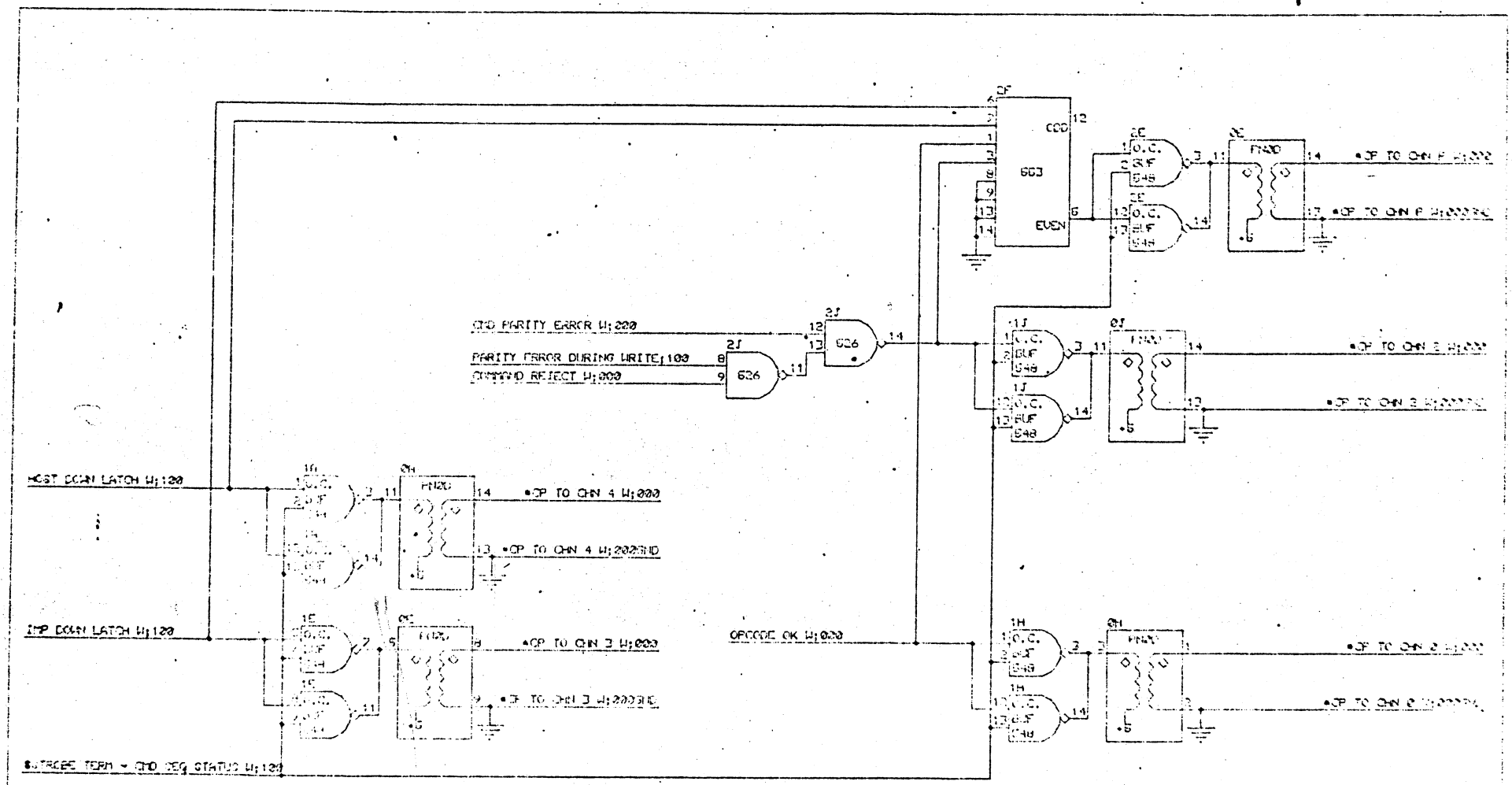
RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI TERM SEQ (WRITE CHAN)	BITS(N-S,) 28-JAN-73	REV



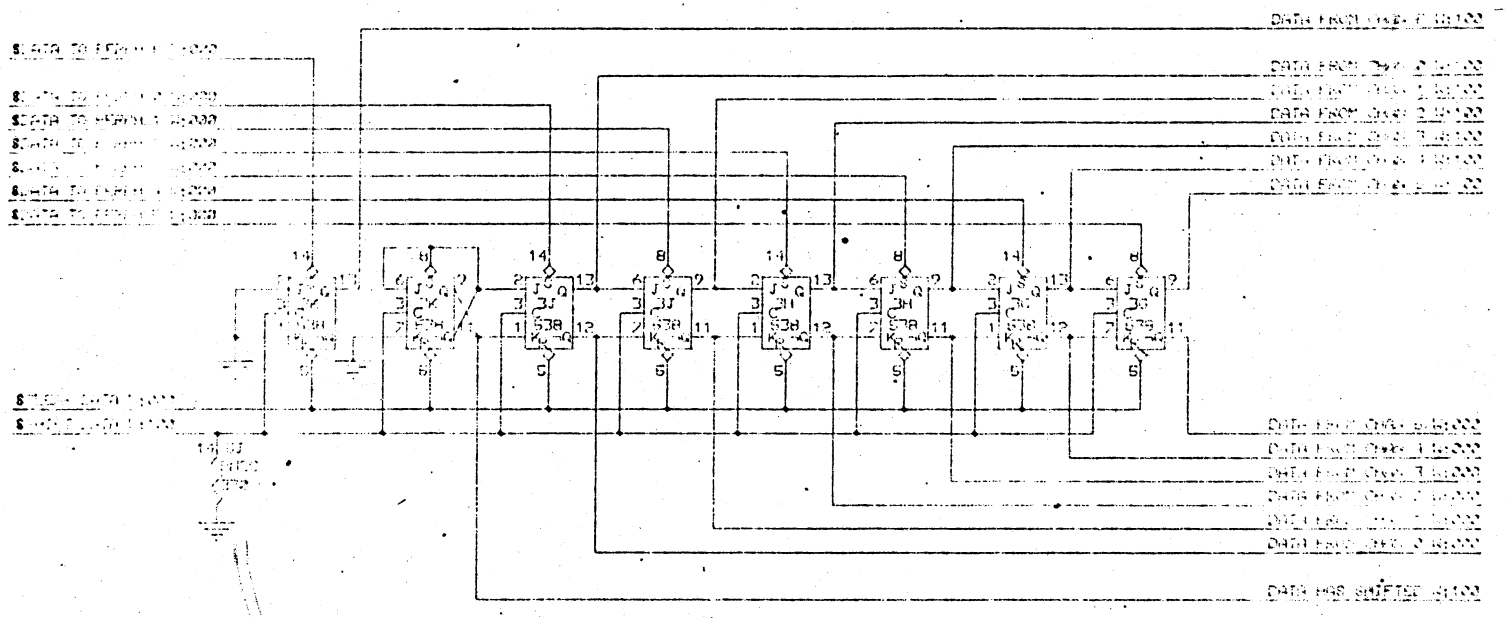
RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI MISC STROBES (WRITE CHANN	
		R-43



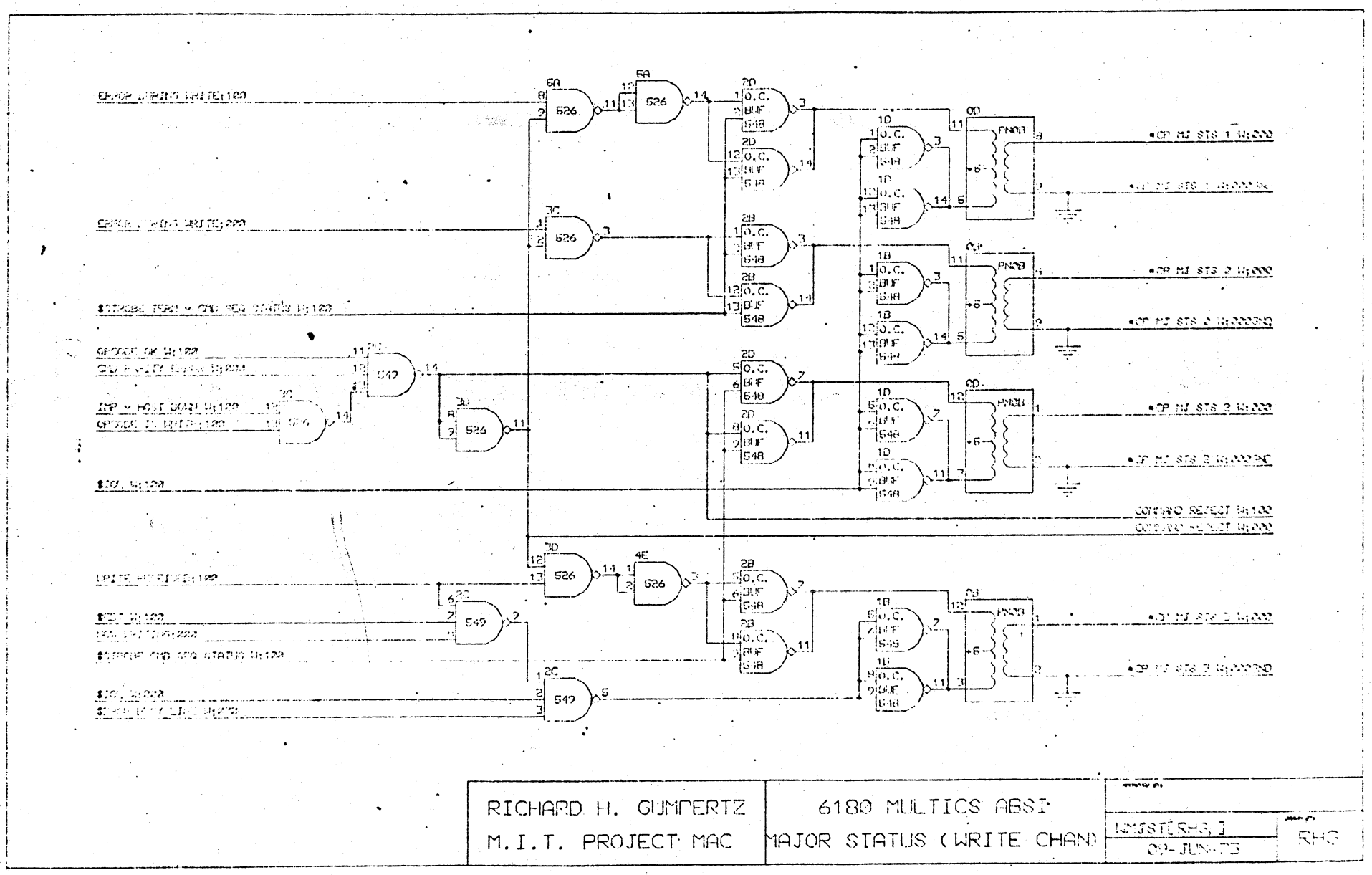
RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI DATA RECVRS (WRITE CHAN)	
		RHG



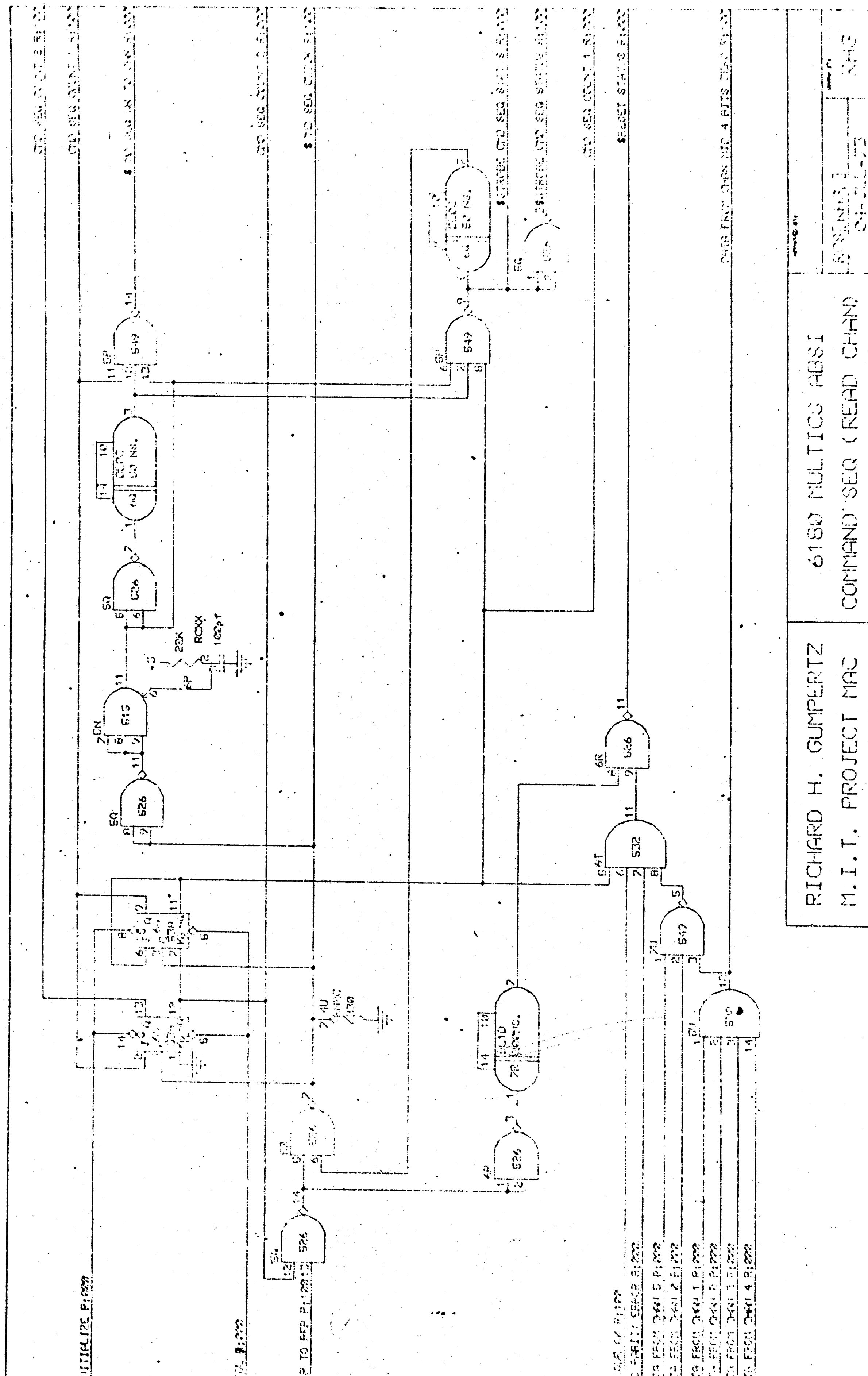
RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI DATA XMITERS (WRITE CHAN)	WDTC[RH3,]	RHG
		04-JUL-73	

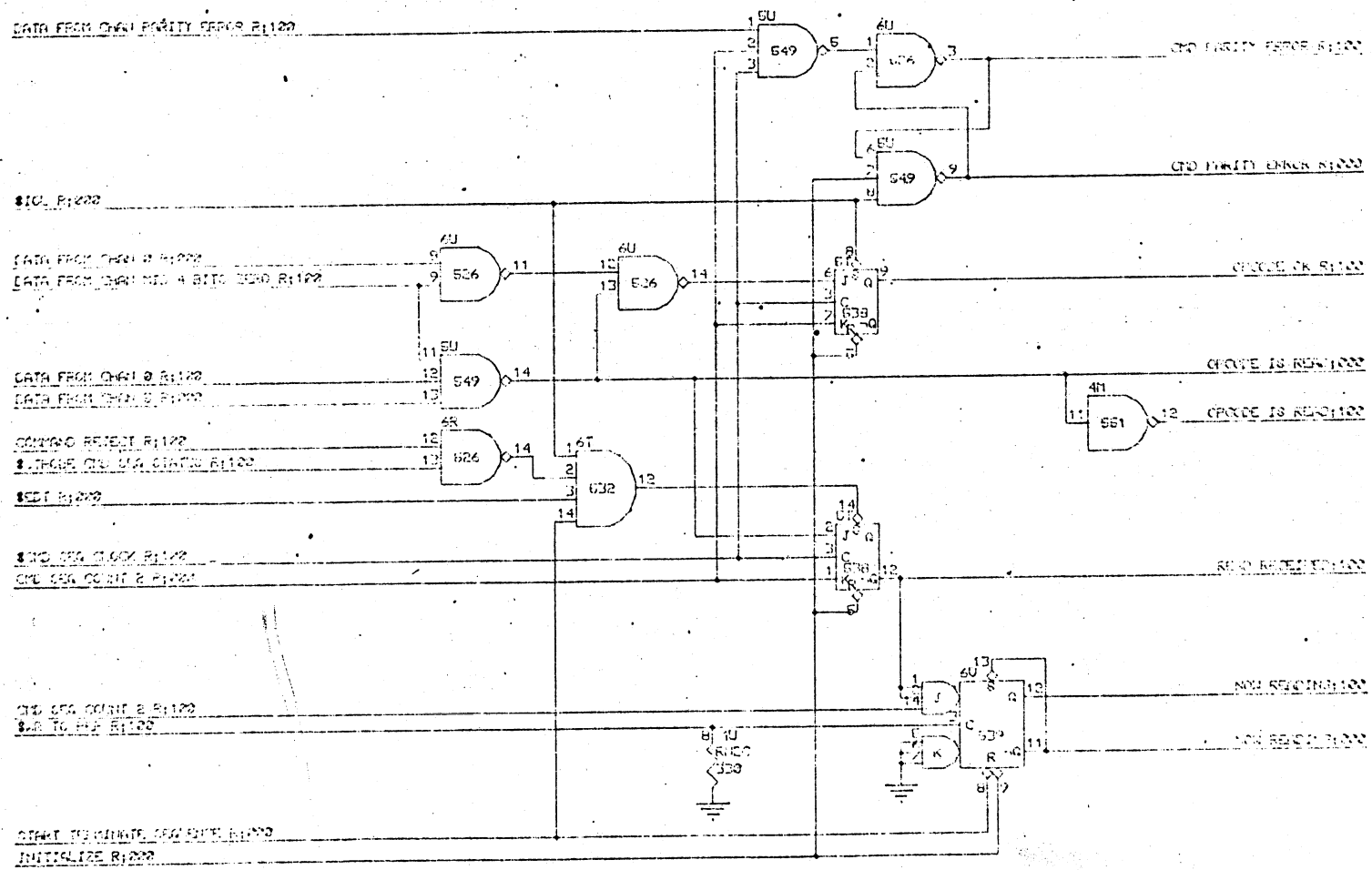


RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI WRITE DATA BUFFER	SIZE B 09-JUN-73

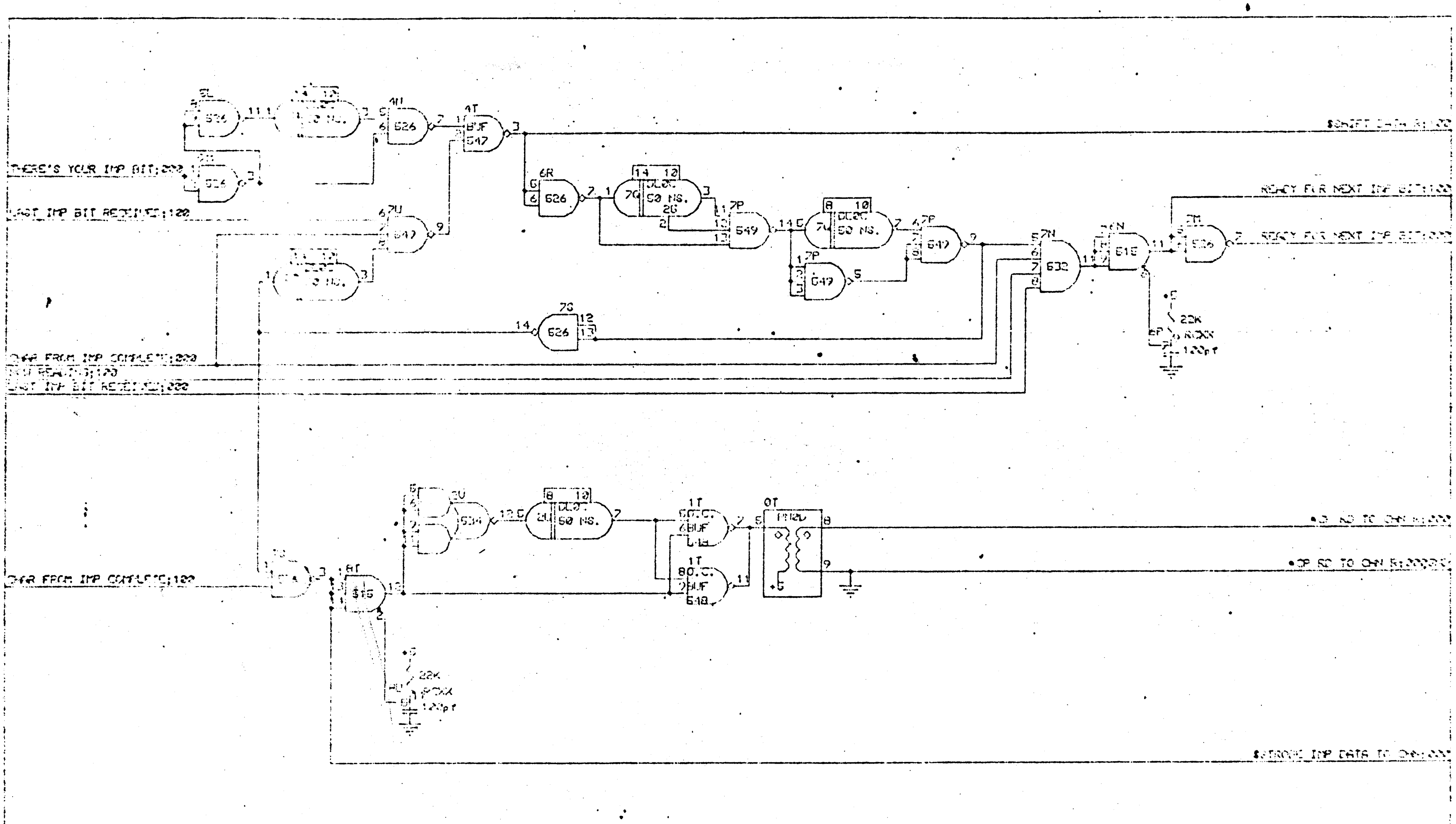


RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI MAJOR STATUS (WRITE CHAN		
		MMJST[RHG.] 09-JUN-73	RHG


RICHARD H. GUMPERTZ
M.I.T. PROJECT MAC
6180 MULTICS ABSI
COMMAND SEQ (READ CHAN)

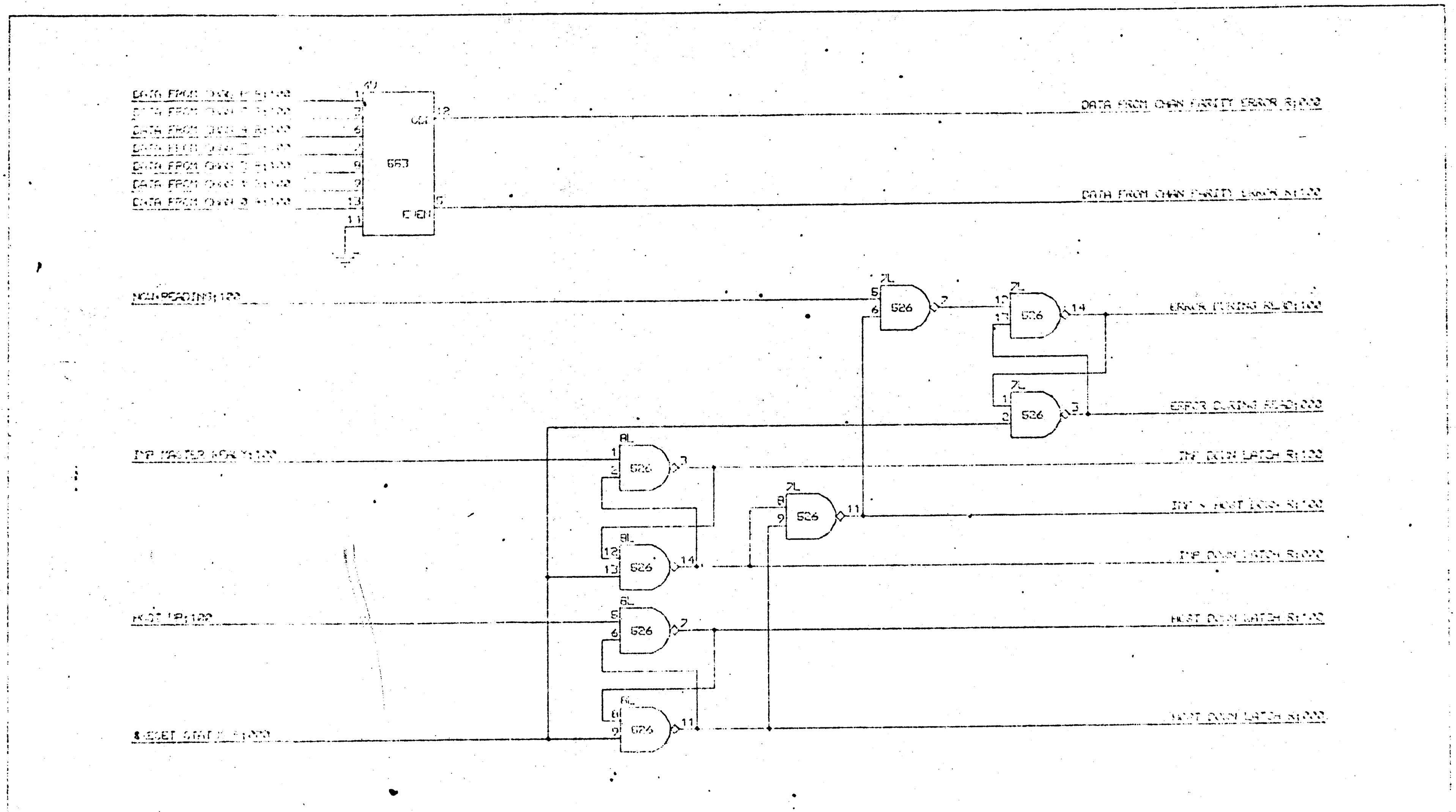


RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI COMMAND CHK (READ CHAN)	ASSEMBLY 04-JUL-73	RHG

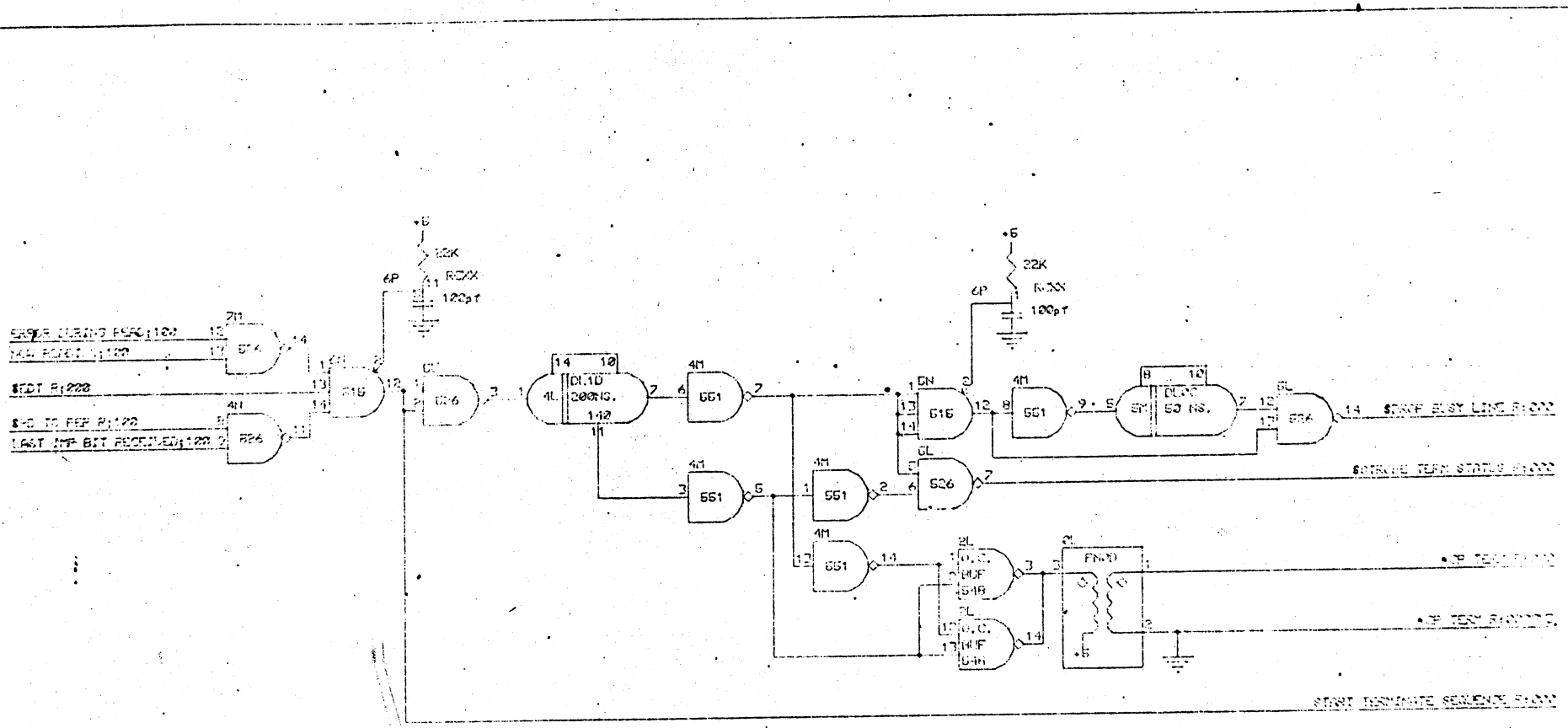


RICHARD H. GUMPERTZ	6180 MULTICS ABSI	RIMFC[RHG,]
M.I.T. PROJECT MAC	IMP CONTROL (READ CHAN)	04-JUL-73

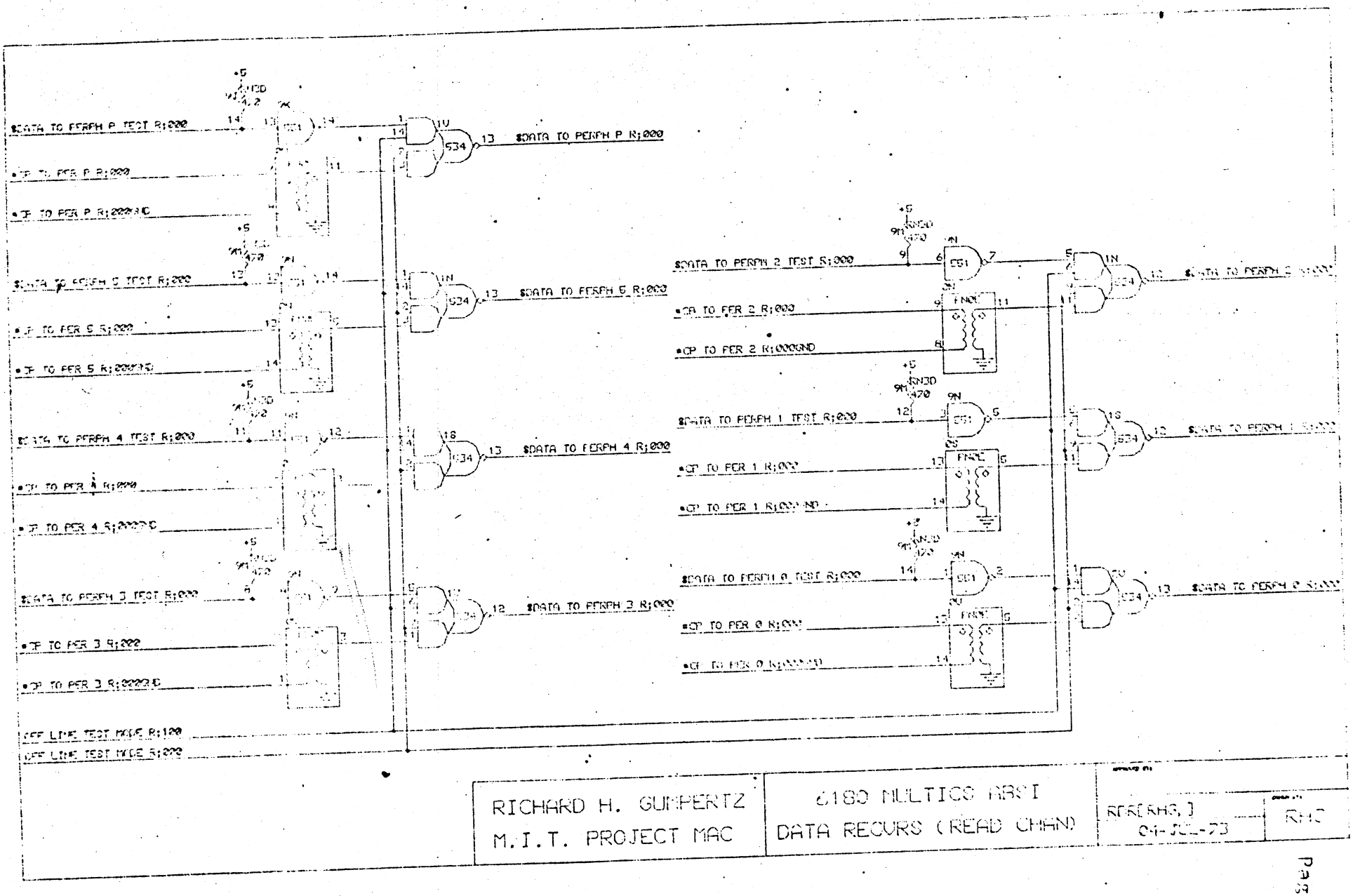
RHG

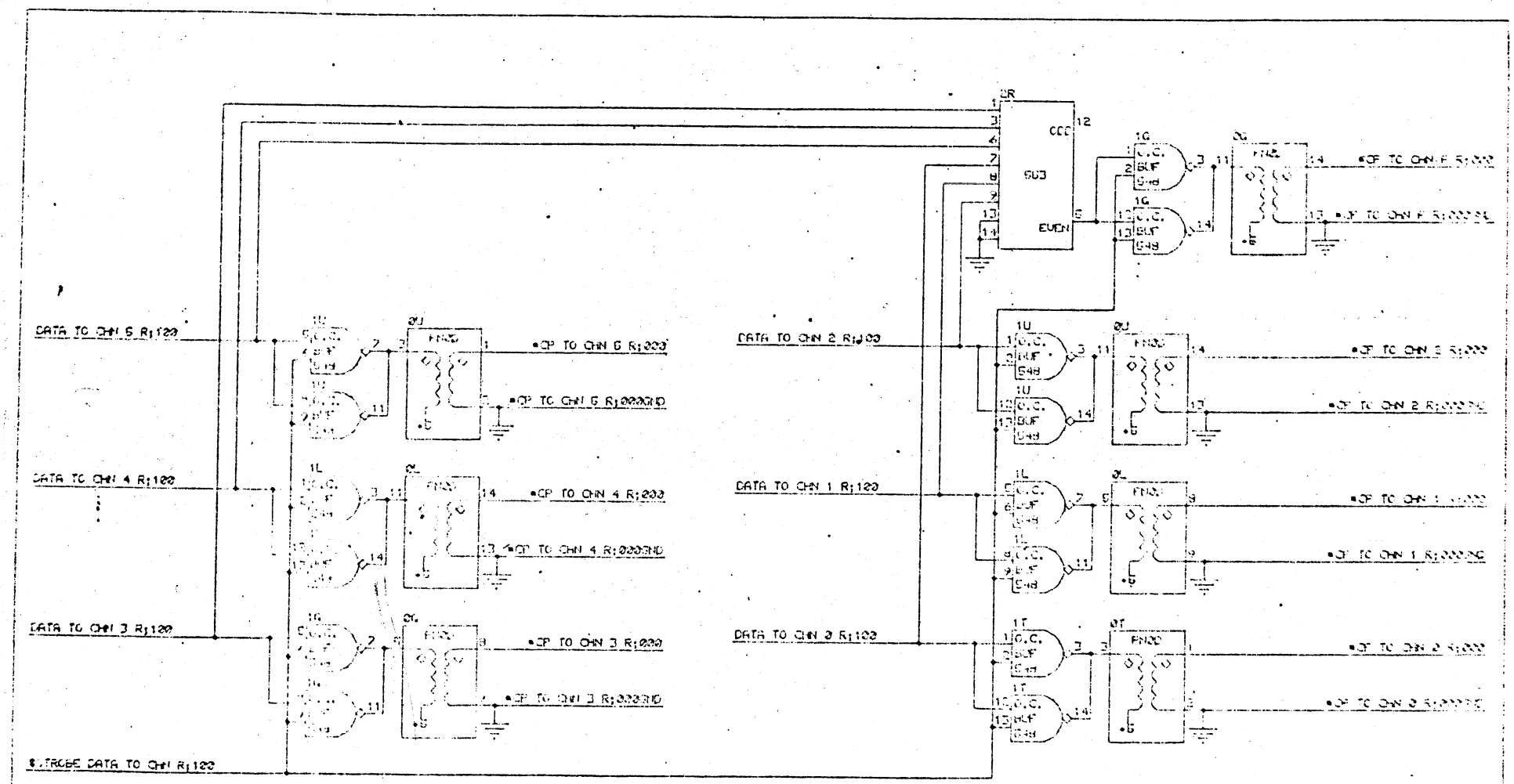


RICHARD H. GUMPERTZ	6180 MULTICS ABSI	REC[RHG,]
M.I.T. PROJECT MAC	ERROR CHECK (READ CHAN)	06-JUN-73

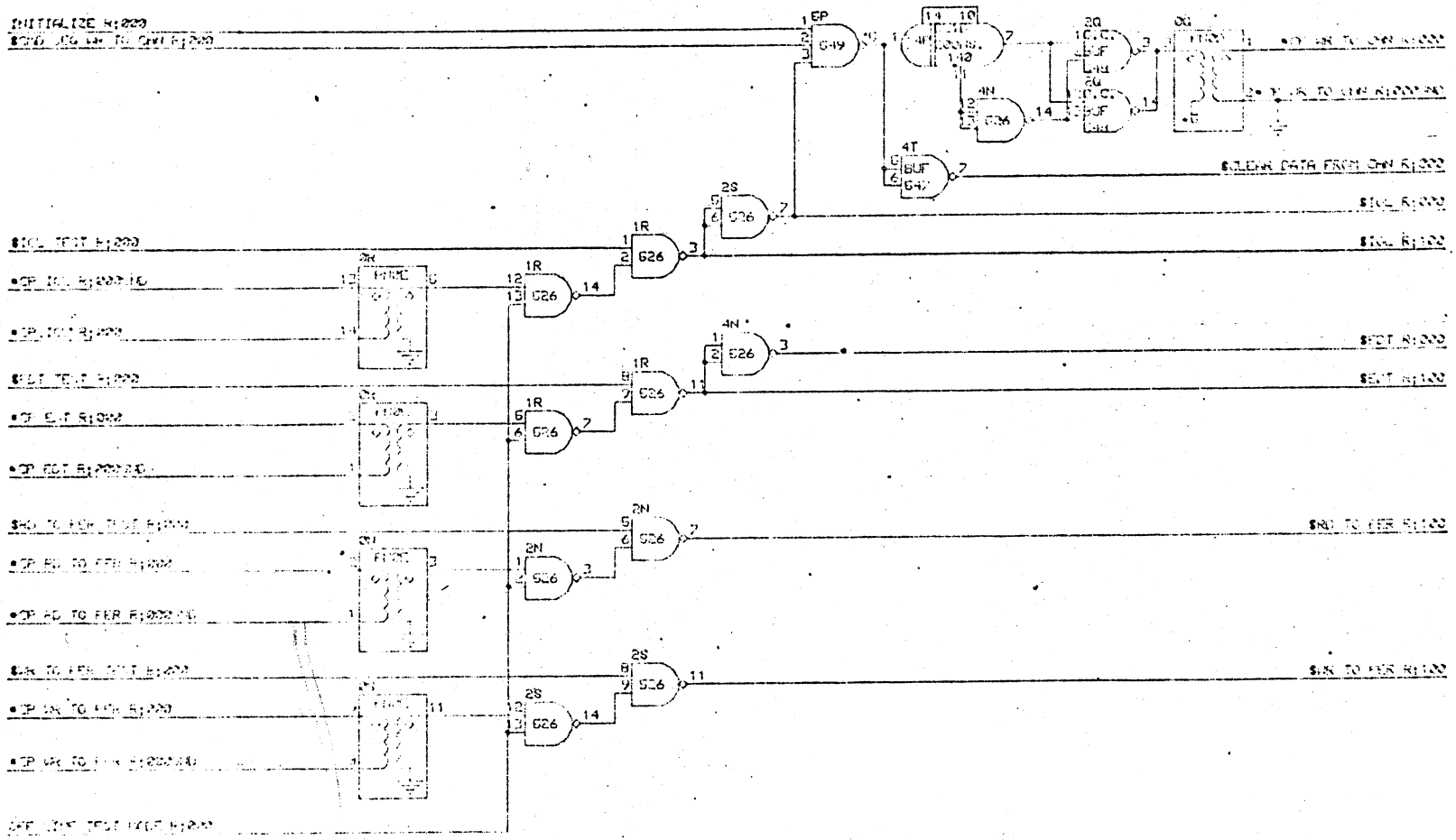


RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI TERM SEQ (READ CHAN)	RTS[RH3.] 04-JUL-73	RH3

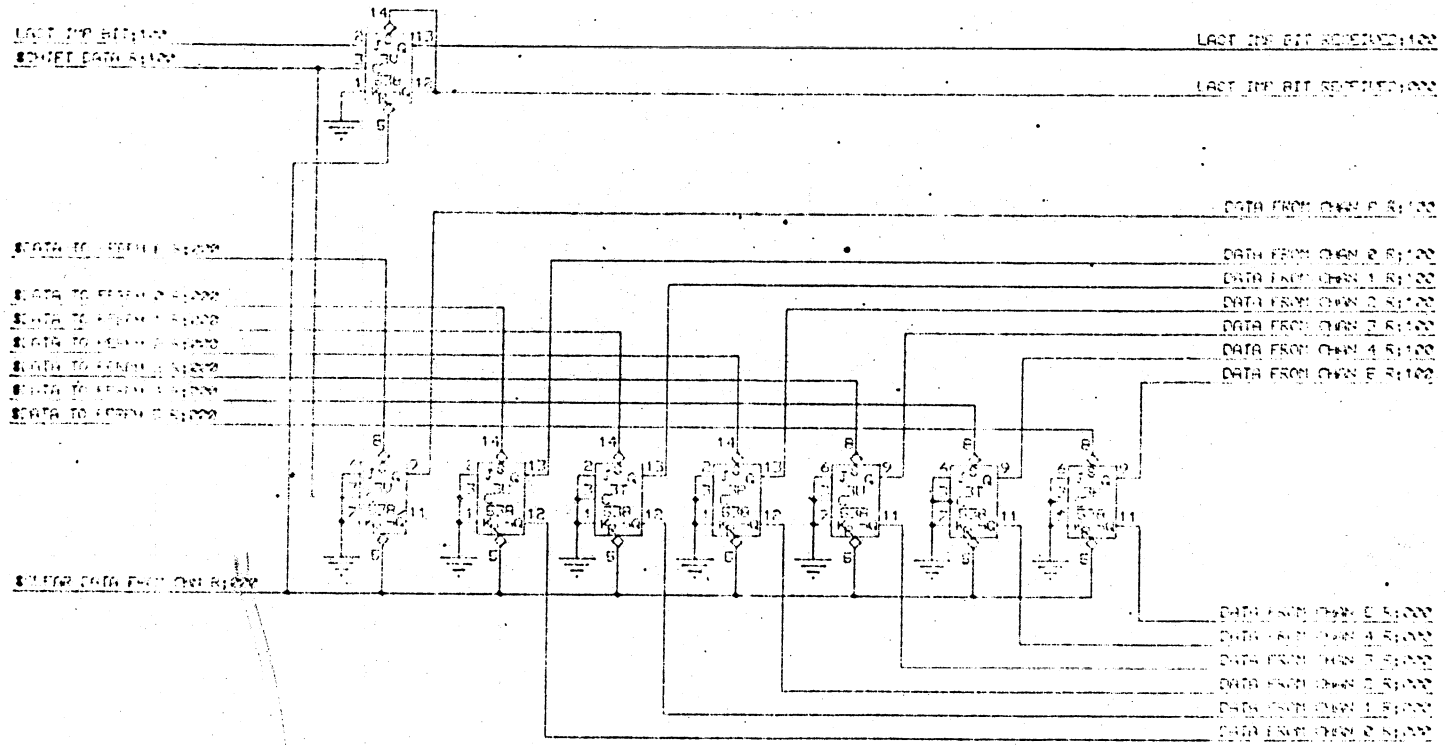

$DATA TO PERPH P TEST R;000
$DATA TO PERPH P R;000
$DATA TO PERPH 5 TEST R;000
$DATA TO PERPH 5 R;000
$DATA TO PERPH 4 TEST R;000
$DATA TO PERPH 4 R;000
$DATA TO PERPH 3 TEST R;000
$DATA TO PERPH 3 R;000
$DATA TO PERPH 2 TEST R;000
$DATA TO PERPH 1 TEST R;000
$DATA TO PERPH 0 TEST R;000
PER LINE TEST MODE R;100
PER LINE TEST MODE R;000
RICHARD H. GUMPERTZ
M.I.T. PROJECT MAC
6180 MULTICS ARSI
DATA RECVRS (READ CHAN)
04-JUL-73
RHG



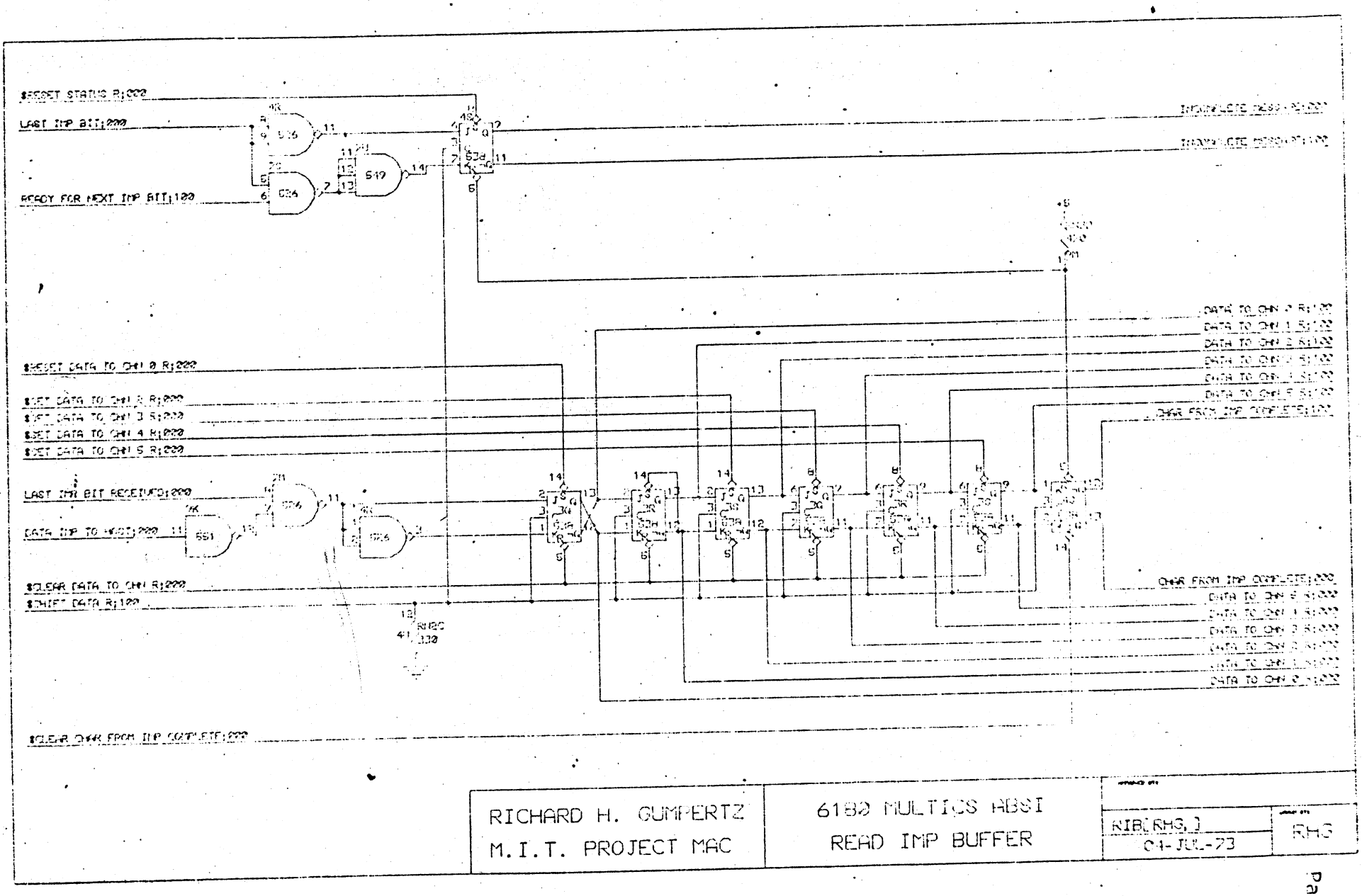
RICHARD H. GUMPERTZ	6180 MULTICS ABSI	
M.I.T. PROJECT MAC	DATA XMITTERS (READ CHAN)	RPT[RHG] 31-JUL-73 / RHG



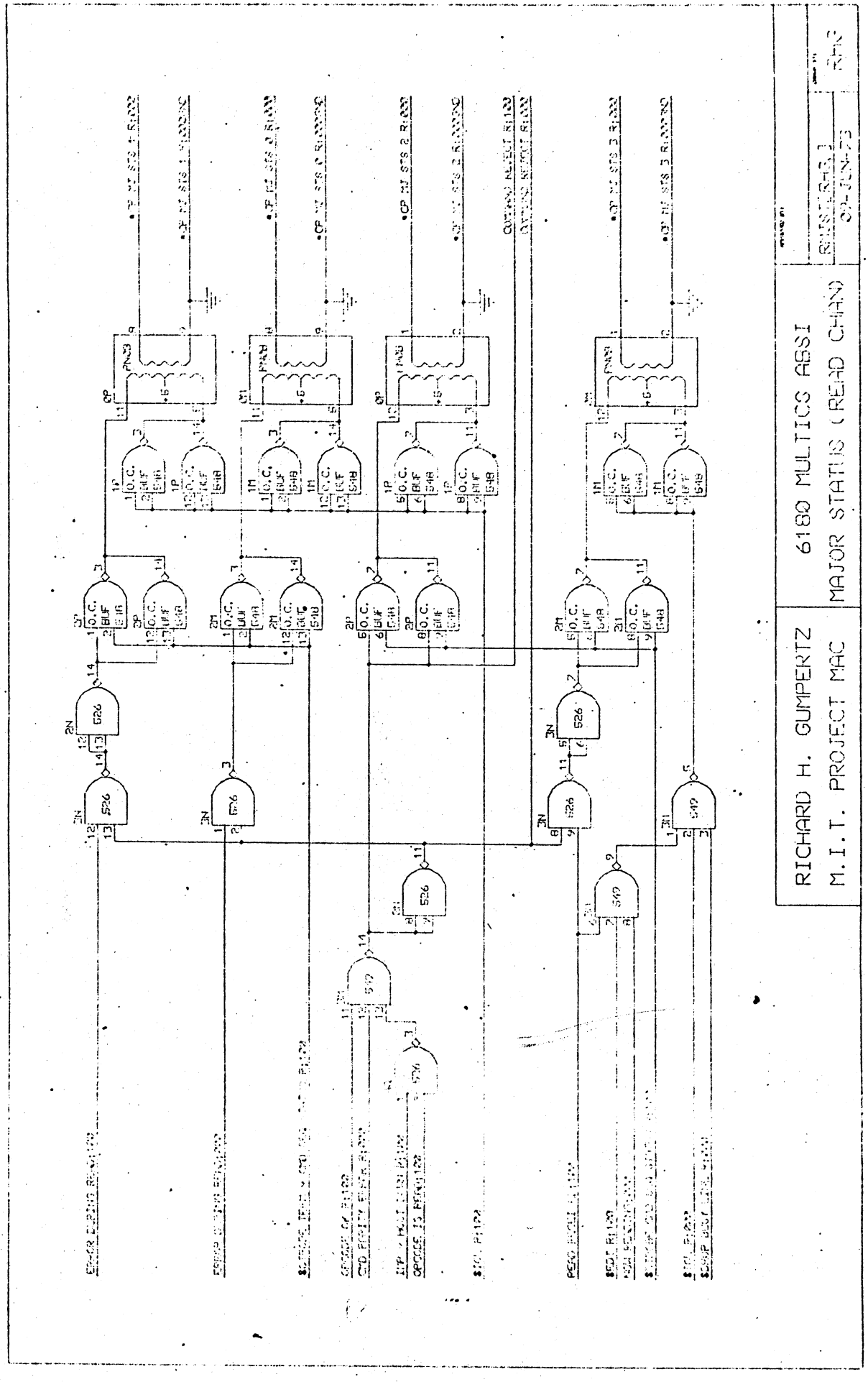
RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI MISC STROBES (READ CHAN)	[illegible] 04-JUL-72

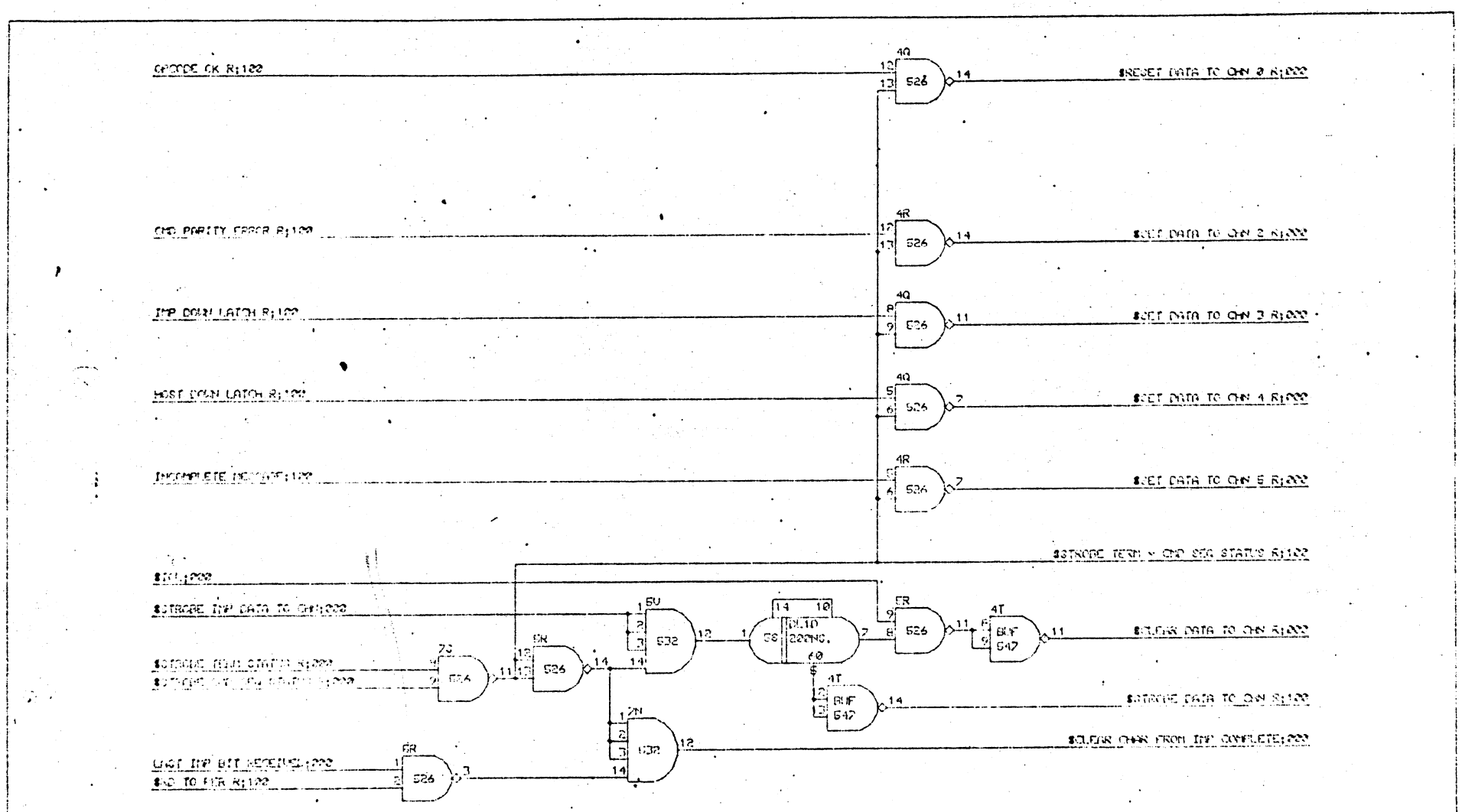


RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI READ COMMAND BUFFER	[illegible] [illegible] 04-30-73	[illegible] R-13

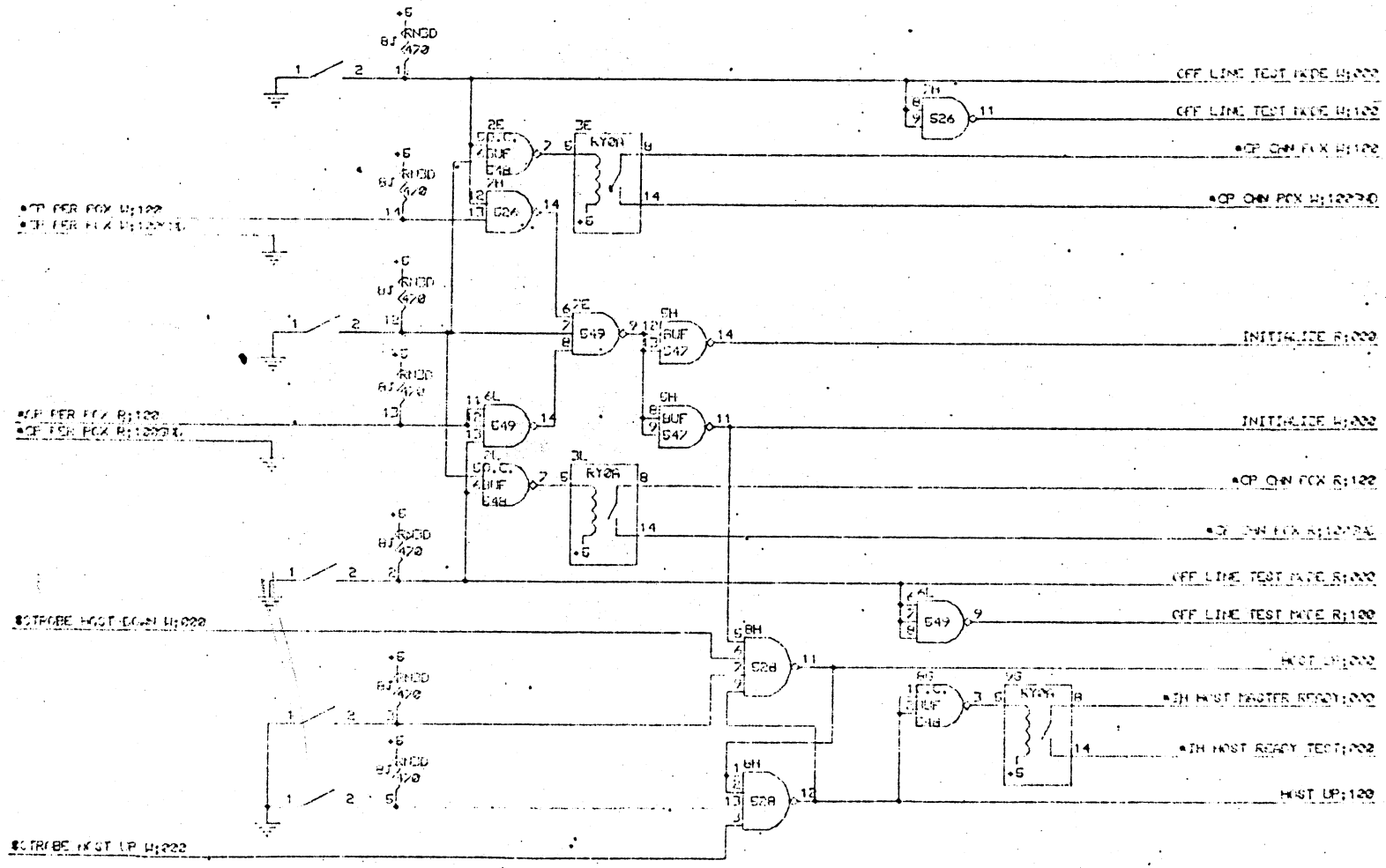


RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS HBSI READ IMP BUFFER	NIB[RHG,] 04-JUL-73	RHG

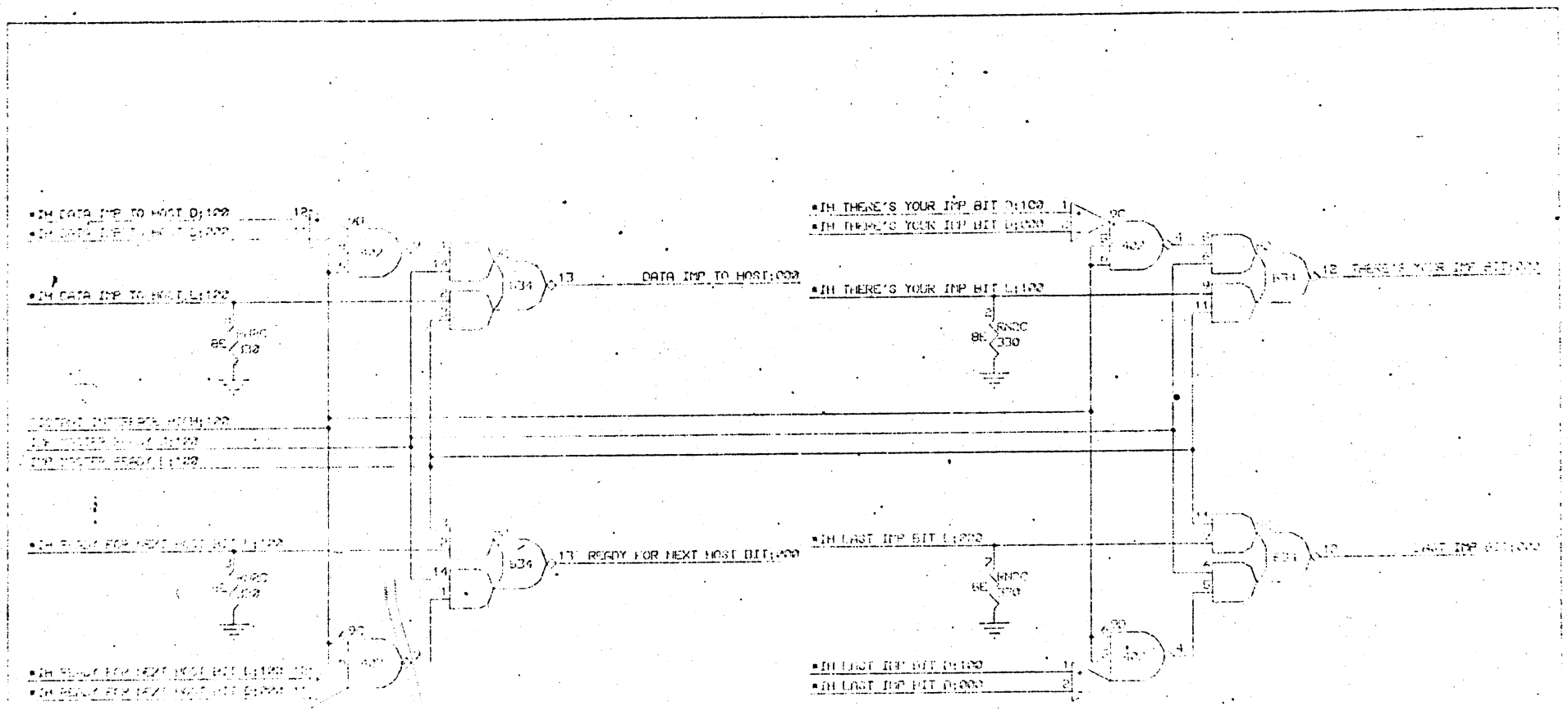

RICHARD H. GUMPERTZ
M.I.T. PROJECT MAC
6180 MULTICS ASSI
MAJOR STATUS (READ CHAN



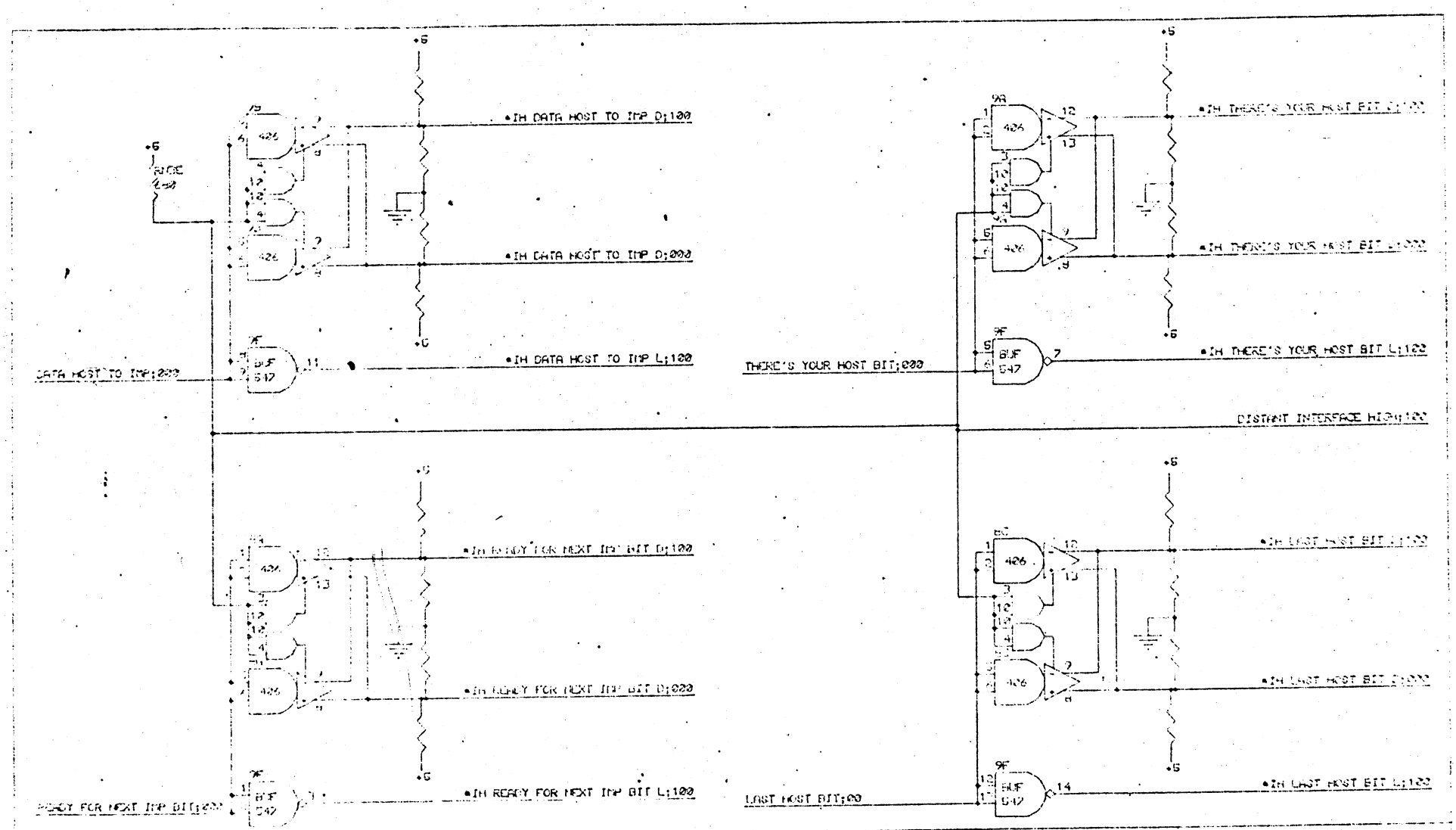
RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI. SUBSTATUS (READ CHAN)	R32[RHG,]	RHG
		04-JUL-73	



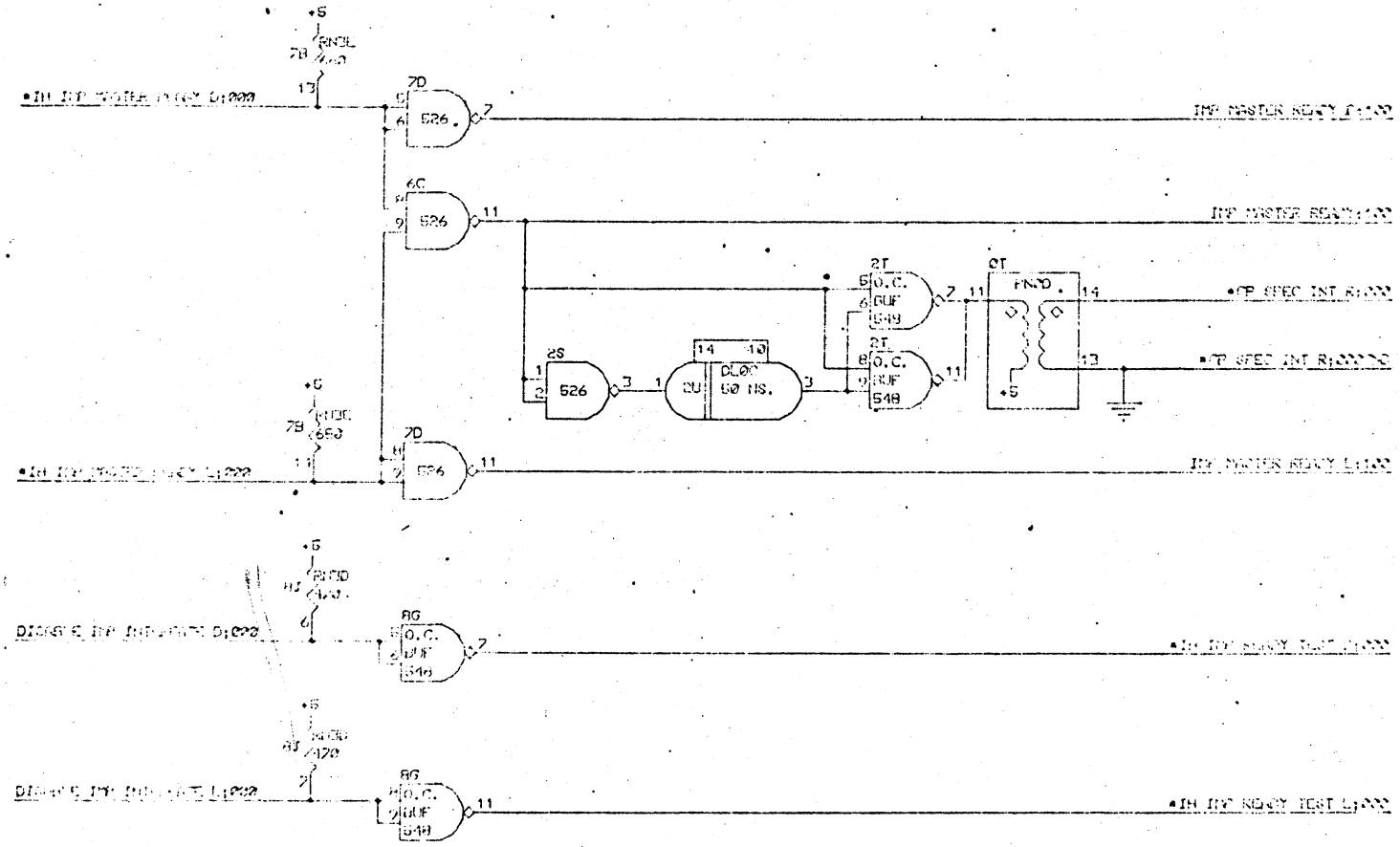
RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI' INITIALIZATION, ETC.	INIT[RHG,] 04-JUL-73	RHG



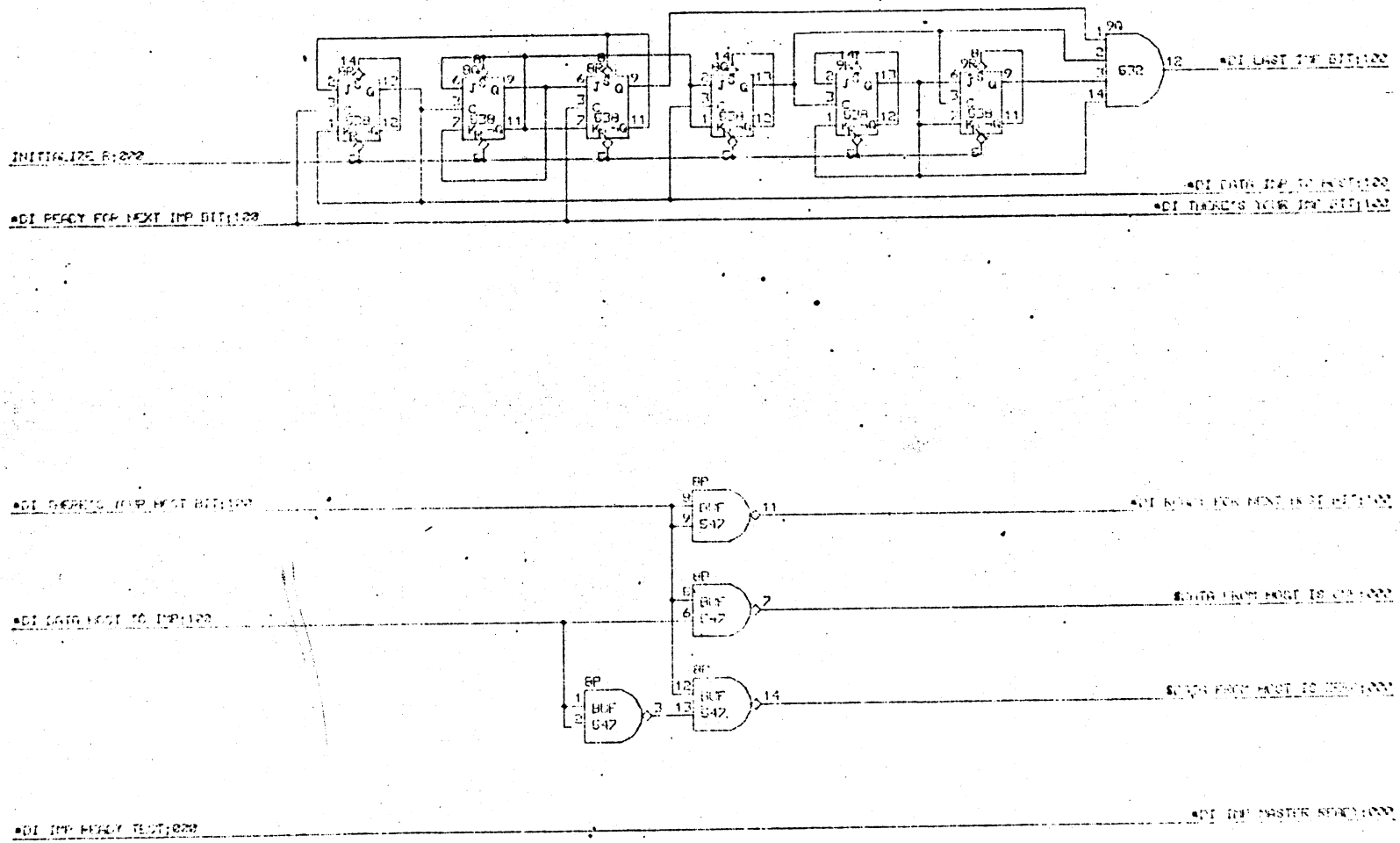
RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ARSI IMP DATA RECEIVERS.		



RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI IMP DATA TRANSMITTERS	IDT[RHG,] 04-JUL-73	RHG



RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ABSI IMP READY CHECK	INST RM3.1 03-JUN-73 RING



RICHARD H. GUMPERTZ M.I.T. PROJECT MAC	6180 MULTICS ARSI IMP SIMULATOR	IMPSIM[RHG,] 18-JUN-73	RHG

Appendix 3

This appendix consists of an excerpt from the signal cross-reference produced automatically on the PDP-10. The omitted portions would probably be of interest only to those involved in maintaining the ABSI.

SIGNAL NAME / LOC(PIN#)	TYPE	LOW	HI	USE	DIPTYPE	BODY	FILE	POS	SLICE
CMD SEQ COUNT 1 R;000									
5P(8)	IS	1.50	0.04	B:IN3	HW549	549	RCS	C3	
6S(6)	I	1.00	0.04	B:J	HW538	538	RCS	A2	
6S(11)	O	13.50	0.80	B:Q0	HW538	538	RCS	A2	
6T(5)	IS	1.50	0.04	B:IN1	HW532	532	RCS	D2	
3 IN 1	OUT	4.00/13.50 LOADING							
CMD SEQ COUNT 1 R;100									
5P(11)	IS	1.50	0.04	C:IN1	HW549	549	RCS	A3	
6S(2)	I	1.00	0.04	A:J	HW538	538	RCS	A1	
6S(9)	O	13.50	0.80	B:Q1	HW538	538	RCS	A2	
2 IN 1	OUT	2.50/13.50 LOADING							
CMD SEQ COUNT 1 W;000									
3F(1)	IS	1.50	0.04	A:IN1	HW532	532	WCS	C2	
5G(6)	I	1.00	0.04	B:J	HW538	538	WCS	A2	
5G(11)	O	13.50	0.80	B:Q0	HW538	538	WCS	A2	
6G(8)	IS	1.50	0.04	B:IN3	HW549	549	WCS	C3	
3 IN 1	OUT	4.00/13.50 LOADING							
CMD SEQ COUNT 1 W;100									
4G(1)	IS	1.50	0.04	A:IN1	HW549	549	WCS	A3	
5G(2)	I	1.00	0.04	A:J	HW538	538	WCS	A1	
5G(9)	O	13.50	0.80	B:Q1	HW538	538	WCS	A2	
2 IN 1	OUT	2.50/13.50 LOADING							
CMD SEQ COUNT 2 R;000									
6S(7)	I	1.00	0.04	B:K	HW538	538	RCS	A2	
6S(12)	O	13.50	0.80	A:Q0	HW538	538	RCS	A1	
5T(1)	I	1.00	0.04	A:K	HW538	538	RCC	C3	
5T(7)	I	1.00	0.04	B:K	HW538	538	RCC	B3	
5U(2)	IS	1.50	0.04	A:IN2	HW549	549	RCC	A3	
5Q(12)	IS	1.50	0.04	D:IN1	HW526	526	RCS	B1	
5 IN 1	OUT	6.00/13.50 LOADING							
CMD SEQ COUNT 2 R;100									
6S(13)	O	13.50	0.80	A:Q1	HW538	538	RCS	A1	
6V(14)	IS	1.00	0.04	J3	HW539	539	RCC	D3	
1 IN 1	OUT	1.00/13.50 LOADING							
CMD SEQ COUNT 2 W;000									
6F(5)	IS	1.50	0.04	B:IN1	HW526	526	WCS	D1	
6H(1)	I	1.00	0.04	A:K	HW538	538	WCC	B2	
6H(7)	I	1.00	0.04	B:K	HW538	538	WCC	C2	
5K(2)	IS	1.50	0.04	A:IN2	HW549	549	WCC	A2	
5G(7)	I	1.00	0.04	B:K	HW538	538	WCS	A2	
5G(12)	O	13.50	0.80	A:Q0	HW538	538	WCS	A1	
5 IN 1	OUT	6.00/13.50 LOADING							

SIGNAL NAME / LOC(PIN#)	TYPE	LOW	41	USE	DIPTYPE	BODY	FILE	POS	SLICE
CMD SEq COUNT 2 W;100									
5G(13)	O	13.50	0.80	A:Q1	HW538	538	WCS	A1	
7G(14)	IS	1.00	0.04	J3	HW539	539	WCC	D3	
1 IN 1	OUT	1.00/13.50	LOADING						
COMMAND REJECT R;000									
2N(11)	O	13.50	0.80	C:OUT	HW526	526	RMJST	C2	
3N(2)	IS	1.50	0.04	A:IN2	HW526	526	RMJST	A2	
3N(8)	IS	1.50	0.04	C:IN1	HW526	526	RMJST	C2	
3N(13)	IS	1.50	0.04	D:IN2	HW526	526	RMJST	A2	
3 IN 1	OUT	4.50/13.50	LOADING						
COMMAND REJECT R;100									
3M(14)	O	13.50	0.80	C:OUT	HW549	549	RMJST	B1	
2N(8)	IS	1.50	0.04	C:IN1	HW526	526	RMJST	C2	
2N(9)	IS	(1.50)	0.04	C:IN2	HW526	526	RMJST	C2	
2P(5)	IS	1.50	0.04	B:IN1	HW548	548	RMJST	B3	
2P(8)	IS	1.50	0.04	C:IN1	HW548	548	RMJST	C3	
6R(12)	IS	1.50	0.04	D:IN1	HW526	526	RCC	B2	
5 IN 1	OUT	6.00/13.50	LOADING						
COMMAND REJECT W;000									
5A(9)	IS	1.50	0.04	C:IN2	HW526	526	WMJST	A2	
3C(2)	IS	1.50	0.04	A:IN2	HW526	526	WMJST	A2	
3D(11)	O	13.50	0.80	C:OUT	HW526	526	WMJST	C2	
3D(12)	IS	1.50	0.04	D:IN1	HW526	526	WMJST	C2	
2J(9)	IS	1.50	0.04	C:IN2	HW526	526	WDT	B2	
4 IN 1	OUT	6.00/13.50	LOADING						
COMMAND REJECT W;100									
2C(14)	O	13.50	0.80	C:OUT	HW549	549	WMJST	B1	
2D(5)	IS	1.50	0.04	B:IN1	HW548	548	WMJST	B3	
2D(8)	IS	1.50	0.04	C:IN1	HW548	548	WMJST	C3	
3D(8)	IS	1.50	0.04	C:IN1	HW526	526	WMJST	C2	
3D(9)	IS	(1.50)	0.04	C:IN2	HW526	526	WMJST	C2	
6K(1)	IS	1.50	0.04	A:IN1	HW526	526	WCC	B1	
5 IN 1	OUT	6.00/13.50	LOADING						
DATA FROM CHAN 0 R;000									
3U(12)	O	13.50	0.80	A:Q0	HW538	538	RCB	C2	
6U(8)	IS	1.50	0.04	C:IN1	HW526	526	RCC	B2	
7U(2)	IS	1.50	0.04	A:IN2	HW549	549	RCS	D2	
2 IN 1	OUT	3.00/13.50	LOADING						
DATA FROM CHAN 0 R;100									
3U(13)	O	13.50	0.80	A:Q1	HW538	538	RCB	C2	
5U(12)	IS	1.50	0.04	C:IN2	HW549	549	RCC	B2	
4V(13)	I	0.75	0.04	IN7	HW553	553	REC	A1	
2 IN 1	OUT	2.25/13.50	LOADING						

SIGNAL NAME LOC(PIN#)	TYPE	LOW	HI	USE	DIPTYPE	BODY	FILE	POS	SLICE
DATA FROM CHAN 0 W;000									
4H(2)	IS	1.50	0.04	A:IN2	HW528	528	WCC	A1	
3J(7)	I	1.00	0.04	B:K	HW538	538	WDB	B2	
3J(12)	O	13.50	0.80	A:Q0	HW538	538	WDB	B2	
4G(12)	IS	1.50	0.04	C:IN2	HW549	549	WIMPC	C1	
3 IN 1 OUT		4.00/13.50 LOADING							
DATA FROM CHAN 0 W;100									
3J(6)	I	1.00	0.04	B:J	HW538	538	WDB	B2	
3J(13)	O	13.50	0.80	A:Q1	HW538	538	WDB	B2	
4J(5)	IS	1.50	0.04	IN4	HW530	530	WCC	B1	
4J(6)	IS	(1.50)	0.04	IN5	HW530	530	WCC	B1	
4K(13)	I	0.75	0.04	IN7	HW553	553	WEC	A1	
4 IN 1 OUT		3.25/13.50 LOADING							
DATA FROM CHAN 1 R;000									
3T(12)	O	13.50	0.80	A:Q0	HW538	538	RCB	C2	
5V(1)	IS	1.50	0.04	A:IN1	HW532	532	RCS	D2	
1 IN 1 OUT		1.50/13.50 LOADING							
DATA FROM CHAN 1 R;100									
3T(13)	O	13.50	0.80	A:Q1	HW538	538	RCB	C2	
4V(9)	I	0.75	0.04	IN6	HW553	553	REC	A1	
1 IN 1 OUT		0.75/13.50 LOADING							
DATA FROM CHAN 1 W;000									
3H(1)	I	1.00	0.04	A:K	HW538	538	WDB	B3	
4H(13)	IS	1.50	0.04	A:IN4	HW528	528	WCC	A1	
3J(11)	O	13.50	0.80	B:Q0	HW538	538	WDB	B2	
4J(2)	IS	1.50	0.04	IN2	HW530	530	WCC	B1	
4G(13)	IS	1.50	0.04	C:IN3	HW549	549	WIMPC	C1	
4 IN 1 OUT		5.50/13.50 LOADING							
DATA FROM CHAN 1 W;100									
3H(2)	I	1.00	0.04	A:J	HW538	538	WDB	B3	
3J(9)	O	13.50	0.80	B:Q1	HW538	538	WDB	B2	
4K(9)	I	0.75	0.04	IN6	HW553	553	WEC	A1	
2 IN 1 OUT		1.75/13.50 LOADING							
DATA FROM CHAN 2 R;000									
3P(12)	O	13.50	0.80	A:Q0	HW538	538	RCB	C2	
5V(2)	IS	1.50	0.04	A:IN2	HW532	532	RCS	D2	
1 IN 1 OUT		1.50/13.50 LOADING							
DATA FROM CHAN 2 R;100									
3P(13)	O	13.50	0.80	A:Q1	HW538	538	RCB	C2	
4V(8)	I	0.75	0.04	IN5	HW553	553	REC	A1	
1 IN 1 OUT		0.75/13.50 LOADING							

SIGNAL NAME	LOC(PIN#)	TYPE	LOW	HI	USE	DIPTYPE	BODY	FILE	POS	SLICE
DATA FROM CHAN 2 W;000										
	3H(7)	I	1.00	0.04	B:K	HW538	538	WDB	B3	
	3H(12)	O	13.50	0.80	A:Q0	HW538	538	WDB	B3	
	4H(3)	IS	1.50	0.04	A:IN3	HW528	528	WCC	A1	
	4J(1)	IS	1.50	0.04	IN1	HW530	530	WCC	B1	
	4G(11)	IS	1.50	0.04	C:IN1	HW549	549	WIMPC	C1	
	4 IN	1 OUT	5.50/13.50 LOADING							
DATA FROM CHAN 2 W;100										
	3H(6)	I	1.00	0.04	B:J	HW538	538	WDB	B3	
	3H(15)	O	13.50	0.80	A:Q1	HW538	538	WDB	B3	
	4K(8)	I	0.75	0.04	IN5	HW553	553	WEC	A1	
	2 IN	1 OUT	1.75/13.50 LOADING							
DATA FROM CHAN 3 R;000										
	3U(11)	O	13.50	0.80	B:Q0	HW538	538	RCB	C3	
	5V(3)	IS	1.50	0.04	A:IN3	HW532	532	RCS	D2	
	1 IN	1 OUT	1.50/13.50 LOADING							
DATA FROM CHAN 3 R;100										
	3U(9)	O	13.50	0.80	B:Q1	HW538	538	RCB	C3	
	4V(7)	I	0.75	0.04	IN4	HW553	553	REC	A1	
	1 IN	1 OUT	0.75/13.50 LOADING							
DATA FROM CHAN 3 W;000										
	5B(7)	IS	1.50	0.04	B:IN2	HW549	549	WIMPC	A3	
	6E(7)	I	1.00	0.04	B:K	HW538	538	WIMPC	C3	
	3H(11)	O	13.50	0.80	B:Q0	HW538	538	WDB	B3	
	4H(1)	IS	1.50	0.04	A:IN1	HW528	528	WCC	A1	
	3G(1)	I	1.00	0.04	A:K	HW538	538	WDB	B3	
	6G(1)	IS	1.50	0.04	A:IN1	HW549	549	WIMPC	A3	
	5 IN	1 OUT	6.50/13.50 LOADING							
DATA FROM CHAN 3 W;100										
	3H(9)	O	13.50	0.80	B:Q1	HW538	538	WDB	B3	
	4J(3)	IS	1.50	0.04	IN3	HW530	530	WCC	B1	
	4K(7)	I	0.75	0.04	IN4	HW553	553	WEC	A1	
	3G(2)	I	1.00	0.04	A:J	HW538	538	WDB	B3	
	3 IN	1 OUT	3.25/13.50 LOADING							
DATA FROM CHAN 4 R;000										
	3T(11)	O	13.50	0.80	B:Q0	HW538	538	RCB	C3	
	5V(14)	IS	1.50	0.04	A:IN4	HW532	532	RCS	D2	
	1 IN	1 OUT	1.50/13.50 LOADING							
DATA FROM CHAN 4 R;100										
	3T(9)	O	13.50	0.80	B:Q1	HW538	538	RCB	C3	
	4V(6)	I	0.75	0.04	IN3	HW553	553	REC	A1	
	1 IN	1 OUT	0.75/13.50 LOADING							

SIGNAL NAME	LOC(PIN#)	TYPE	LOW	HI	USE	DIPTYPE	BODY	FILE	POS	SLICE
DATA FROM CHAN 4 W;000										
	4J(7)	IS	1.50	0.04	IN6	HW530	530	WCC	B1	
	4J(9)	IS	(1.50)	0.04	IN7	HW530	530	WCC	B1	
	3G(7)	I	1.00	0.04	B:K	HW538	538	WDB	B4	
	3G(12)	O	13.50	0.80	A:Q0	HW538	538	WDB	B3	
	4G(7)	IS	1.50	0.04	B:IN2	HW549	549	WCS	C2	
	4 IN	1 OUT	4.00/13.50 LOADING							
DATA FROM CHAN 4 W;100										
	3F(7)	IS	1.50	0.04	B:IN3	HW532	532	WCS	D2	
	4A(6)	I	0.75	0.04	IN3	HW553	553	WEC	A1	
	3G(6)	I	1.00	0.04	B:J	HW538	538	WDB	B4	
	3G(13)	O	13.50	0.80	A:Q1	HW538	538	WDB	B3	
	3 IN	1 OUT	3.25/13.50 LOADING							
DATA FROM CHAN 5 R;000										
	3P(11)	O	13.50	0.80	B:Q0	HW538	538	RCB	C3	
	5G(13)	IS	1.50	0.04	C:IN3	HW549	549	RCC	B2	
	7J(1)	IS	1.50	0.04	A:IN1	HW549	549	RCS	D2	
	2 IN	1 OUT	3.00/13.50 LOADING							
DATA FROM CHAN 5 R;100										
	3P(9)	O	13.50	0.80	B:Q1	HW538	538	RCB	C3	
	4V(5)	I	0.75	0.04	IN2	HW553	553	REC	A1	
	1 IN	1 OUT	0.75/13.50 LOADING							
DATA FROM CHAN 5 W;000										
	7H(5)	IS	1.50	0.04	B:IN1	HW526	526	WCS	D3	
	4J(13)	IS	1.50	0.04	IN8	HW530	530	WCC	B1	
	3G(11)	O	13.50	0.80	B:Q0	HW538	538	WDB	B4	
	4G(6)	IS	1.50	0.04	B:IN1	HW549	549	WCS	C2	
	3 IN	1 OUT	4.50/13.50 LOADING							
DATA FROM CHAN 5 W;100										
	7H(1)	IS	1.50	0.04	A:IN1	HW526	526	WCS	D3	
	4A(5)	I	0.75	0.04	IN2	HW553	553	WEC	A1	
	3G(9)	O	13.50	0.80	B:Q1	HW538	538	WDB	B4	
	2 IN	1 OUT	2.25/13.50 LOADING							
DATA FROM CHAN LOW 4 BITS ZERO W;100										
	3F(6)	IS	1.50	0.04	B:IN4	HW532	532	WCS	D2	
	4F(3)	O	13.50	0.80	A:OUT	HW526	526	WCC	A3	
	4F(6)	IS	1.50	0.04	B:IN2	HW526	526	WIMPC	D3	
	4G(8)	IS	1.50	0.04	B:IN3	HW549	549	WCS	C2	
	3 IN	1 OUT	4.50/13.50 LOADING							
DATA FROM CHAN MID 4 BITS ZERO R;100										
	5G(11)	IS	1.50	0.04	C:IN1	HW549	549	RCC	B2	
	6G(9)	IS	1.50	0.04	C:IN2	HW526	526	RCC	B2	
	7U(3)	IS	1.50	0.04	A:IN3	HW549	549	RCS	D2	
	5V(12)	O	12.00	0.40	A:OUT	HW532	532	RCS	D2	
	3 IN	1 OUT	4.50/12.00 LOADING							